NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

OCEAN BOTTOM SIMULATION USING FRACTAL GEOMETRY

by

Candace J. Robertson

September 1991

Thesis Advisor: David Canright

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE				
4 PERFORMING ORGANIZATION REPORT NUMBER(S)		5 MONITORING ORGANIZATION REPORT NUMBER(S)		
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)*	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000		7b ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000		
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS *(City, State, and ZIP Code)*		10 SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE *(Include Security Classification)*
OCEAN BOTTOM SIMULATION USING FRACTAL GEOMETRY (U)

12 PERSONAL AUTHOR(S)
Robertson, Candace J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	September 1991	80

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the autho and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	fractal, ocean bottom simulation topographical simulation

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Fractal geometry can simulate natural topography, creating data that can be used in sonar models as realistic ocean bottom features. An algorithm using recursive subdivision, or midpoint replacement, is used to create the fractals. The appearance, statistics, and dimension of the fractal can be controlled through the use of variables. The variables control the initial corner values and the amount that each subdivision can vary from the average of its two initial points. The choice of a random number distribution also affects the final fractal. The statistics, fractal dimension, and appearance of data generated by the fractal algorithm are comparable to real data.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT		21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS		UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL David Canright	22b TELEPHONE *(Include Area Code)* (408)646-2782	22c OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

Approved for public release; distribution is unlimited

Ocean Bottom Simulation Using Fractal Geometry

by

Candace Joanne Robertson

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: Candace Joanne Robertson
Candace Joanne Robertson

Approved by: [illegible]
David Canright, Thesis Advisor

Jo Ellen Wilbur
JoEllen Wilbur, Thesis Advisor

[illegible]
Elan Moritz, Thesis Advisor

[illegible]
Anthony A. Atchley, Chairman, Engineering
Acoustics Academic Committee

ABSTRACT

Fractal geometry can simulate natural topography, creating data that can be used in sonar models as realistic ocean bottom features. An algorithm using recursive subdivision, or midpoint replacement, is used to create the fractals. The appearance, statistics, and dimension of the fractal can be controlled through the use of variables. The variables control the initial corner values and the amount that each subdivision can vary from the average of its two initial points. The choice of a random number distribution also affects the final fractal. The statistics, fractal dimension, and appearance of data generated by the fractal algorithm are comparable to real data.

TABLE OF CONTENTS

I. INTRODUCTION 1

  A. FRACTAL GEOMETRY 2

  B. MODEL 5

    1. Concept 5

    2. Use 6

II. ALGORITHM 8

  A. MATHEMATICAL DESCRIPTION 9

  B. VARIABLES 12

    1. Corner Values 12

    2. Random Number Generation 14

    3. Alteration 16

  C. JUSTIFICATION AND VALIDATION 18

    1. Visual Inspection 19

    2. Statistical Analysis 20

    3. Fractal Dimension 22

III. CONCLUSION AND RECOMMENDATIONS 24

APPENDIX A - Figures 27

APPENDIX B - Fractal Fortran Code 66
Dimension Fortran Code 68

LIST OF REFERENCES 70

BIBLIOGRAPHY 71

INITIAL DISTRIBUTION LIST 73

LIST OF FIGURES

1. Step 1 in Fractal Creation.......................... 27
2. Step 2 in Fractal Creation.......................... 28
3. Corner Values 16, 32, 64, 128....................... 29
4. Corner Values 64, 64, 64, 64........................ 31
5. Effect of Corner Variable on Mean of Fractal.. 33
6. Effect of Corner Variable on Standard Deviation of Fractal................................ 34
7. Gaussian Distribution............................... 35
8. Binomial Distribution............................... 37
9. Uniform Distribution................................ 39
10. Alteration - Constant of 64........................ 41
11. Alteration - Constant of 5......................... 43
12. Alteration - Increasing............................ 45
13. Alteration - Decreasing............................ 47
14. Sand... 49
15. Rough Wall... 51
16. Wet Sand... 53
17. Gravel... 55
18. Fractal Simulation of Sand......................... 57
19. Fractal Simulation of Rough Wall................... 59
20. Fractal Simulation of Wet Sand..................... 61
21. Fractal Simulation of Gravel....................... 63

I. INTRODUCTION

Knowledge of the shape and texture of the ocean bottom is of interest to the sonar community. The form that the bottom takes, both in fine and gross features, affects the image resulting from sonar ensonification. Current sonar models assume a homogeneous flat planar surface or random facet distribution for the ocean bottom. Although this is efficient for computations it is misleading. The final result may correctly model the behavior of a sonar on the target but it will not include the effects, reflection, reverberation, and shadowing, from a topographically correct bottom.

Although the use of real data is optimum it is expensive, difficult, and time consuming to collect. The solution is the simulation of data that behaves similarly to real data. Fractal geometry allows the simulation of the ocean floor. It can create data that has texture that is expected in natural terrain. Natural surfaces and features have been simulated, at least to the satisfaction of the human eye, using fractal geometry.

In recent years fractals have been used to simulate topography, most notably in movies such as "The Last Starfighter" and "Star Trek - The Wrath of Khan". (Pietgen, Saupe, 1988, p.2) The images created with fractals blend into the movie effect and discrimination between real topography and computer generated topography becomes difficult.

The objective of this thesis is to present a method for simulating a selection of ocean bottoms in terms of bottom types. The simulation will be validated by comparison with real data to establish the merit of using the proposed fractal generation method. The algorithm can be incorporated into sonar models to provide a more realistic background representation. The advantage of this approach is the saving in space that results when fractals are generated as needed and not stored.

By using stored parameters any fractal that can be created by this algorithm can be recreated at any time with no loss of information. The disadvantage is the amount of time that is required to recreate the fractal. The time required to create a fractal surface is dependent upon the computer but presently available computers make this algorithm a reasonable addition to other models adding only seconds of run time.

A. FRACTAL GEOMETRY

There are certain shapes that contain infinite levels of detail. These fractals can fill space in ways that are measurably different from traditional geometric objects. The measurement of geometric objects is accomplished with standard units of measurement. The area of a box is stated in square meters, the measurement of the volume of the box is in cubic meters. But a fractal, having a complicated surface, cannot be fully described by traditional measurement methods. In fact, the measured length of a fractal curve is dependent upon the measuring stick.

The usual explanation of this phenomena involves the measurement of a coastline. In the box-counting dimension method equal-sized boxes, or length units, are placed in the area or along the line to be measured. Any unit of measurement can be used but if a kilometer is used as the measuring standard an answer can be calculated. If the standard is decreased, a meter is the measuring standard and the answer is different, larger, because a meter can register the inlets and coves that a kilometer would neglect. It is true again for a measuring standard of a centimeter. More distance is covered by a smaller measurement standard because smaller deviances in the coastline are measurable. (Mandelbrot, 1983, p.27)

This concept can be extrapolated to higher dimensional objects. A measuring standard could be cubic kilometers, cubic meters, or cubic centimeters. These boxes fill the space to be measured and are countable. Again the unit size is arbitrary as long as the boxes are equal-sized.

The count of the necessary number of boxes is the number N. The size of the box is r. As r decreases, N should increase for a fractal and should be approximated by

$$N = k \cdot r^{-D}.$$

D is the dimension and k is a constant, unimportant for determining the dimension. The fractal dimension used to describe the texture or roughness of a fractal can be compared to a Euclidean concept of dimension. (Canright)

Felix Hausdorff is cited by Mandelbrot as best describing the fractal dimension. Hausdorff stated that it is accepted that the length of the perimeter of an N-sided polygon is N multiplied by the length of its side. Each length is raised to its first power since that is the dimension of a straight line in Euclidean geometry. Similarly, the interior of the polygon can be approximated by adding together the number of squares fitted within the polygon times the width of each square, raised to the second power, the dimension of a plane.

A fractal can be measured in units r. These units can be raised to the Dth power. (Mandelbrot,1977,p.34) The dimension D can be, but is not necessarily, an integer. The non-integer dimension exists somewhere between the dimensions that we can easily visualize. A dimension of 1.8 is more than a straight line but less than a flat plane. The curve of dimension 1.8 fills more space than the curve of dimension one but less than an area of dimension two.

The dimension, D, gives a quantitative measure of the extent of the curve. This is difficult to visualize since traditional geometry still defines the curve with a fractal dimension greater than one but less than two to have a topological dimension of one, topological dimensions always being integers. Many natural shapes can be well characterized by fractals: metal grains, crystals, sand, dust, cauliflowers, trees, and ferns. Geographical fractals describe topographical objects such as lakes, islands and coastlines, all of which can be simulated with fractals.

B. MODEL

The ocean bottom is typified by many sediments: sand, mud, rock, coral, gravel, and combinations. These different media have been studied as to acoustical penetration, diameter of individual particles, and movement caused by underwater wave action. Fractal processes are a good model for natural phenomenon and, as a naturally occurring phenomenon, the ocean bottom can be simulated by fractals. The intent of the fractal simulation is to represent the topography of the ocean bottom using a scaling factor to indicate relative heights of areas on the ocean bottom.

1. Concept

The simulation of topographical features using fractals has been used by Dietmar Saupe (Pietgen, Saupe, 1988, p2), Mandelbrot (Mandelbrot, 1983, Plate C9-C15), and others. The concept is not new but its application to the ocean bottom presumes that the floor of the ocean has similar construction found on the topography above the ocean surface. Different types of ocean sedimentation can be simulated by altering the variables in the algorithm. Smooth surfaces to excessively rough surfaces can be simulated as well as varying elevations above the ocean bottom.

2. Use

The intent of the model is to provide simulation of portions of the ocean floor in studying navigation, image processing, and sonar signal processing. The resulting data will provide information that is otherwise inaccessible for other sonar models. Simulations that attempt to model the performance of a sonar need data that imitates the ocean floor as background to provide a realistic setting. Real ocean floor data is very expensive to obtain and requires vast amounts of storage capacity. Where it does exist the measurements do not extend to the resolution of interest. Measurements are done on a gross scale. Those indicating topology in small increments, even at one square meter are not measured or recorded.

A reasonable simulation of the ocean bottom will allow the validation of sonar models by comparison of actual sonar data with output of the sonar model. Presently sonar models use flat planes or simple geometric descriptions of random number distributions to describe the ocean bottom.

Targets of interest are overlaid onto the background. The man-made target is the point of interest but interactions between the target and a realistic environment are lost.

II. ALGORITHM

The algorithm used to create the fractal bottom simulation is a variation of the recursive subdivision method. This method, also known as the random midpoint displacement method, operates by creating midpoint values using input from points surrounding the selected position and random numbers used to influence the existing position values. In order to demonstrate the influence of each variable the values for all variables are held constant except the variable being discussed. Additionally the seed which generates the random number is held constant.

The fractal images generated differ only where they are affected by the change in the variable being discussed. The mean grey level of the fractal is influenced by the predetermined corner values. The structure and large-scale shape of the fractal is determined by the random number seed. RN determines the distribution in the 256 level grey scale of the pixel values. The texture is determined by the alteration, its size and level placement.

Although these fractals can be said to be locally nondeterministic, or at least unpredictable as to pixel value and placement, these variables allow some element of control as to fractal mean and roughness.

Another tool used to control the fractal is the use of clipping. After the algorithm is completed all values above 255 and below 0 are clipped. The result is comparable to rocks surrounded by drifted sand, or mud, or flat level sea mounts. The reason for this clipping is to force the pixel values into the range 0 to 255 for display and one byte storage. An alternative is to take the full range of values and scale them into a 0 to 255 range. This method would maintain the lowest and highest levels. Clipping can also be employed at each level during the fractal creation. The fractal would be controlled, never allowed to vary too far outside the 0 to 255 range.

A. MATHEMATICAL DESCRIPTION

The algorithm developed for this work generates a two-dimensional array of integers of values between 0 and 255. These can be interpreted as elevations, with the lowest value corresponding to 0, and the highest value to 255. The algorithm starts with four corner values that have been pre-assigned. These values fall between the range of 0 and 255. This range is used throughout this work for two reasons. When displaying the resulting image most displays use a 256 grey scale. This value also stores conveniently as a byte of data for each point of the image providing efficient memory requirements.

A point midway between two corners, horizontally or vertically, is computed using

$$midpoint\,pixel = \frac{pixel\,1 + pixel\,2}{2} + RN \cdot alteration\ .$$

RN is a random number between -1 and 1. The random number is generated on the computer by its random number generator. Although the computer actually generates a pseudorandom number it is sufficiently random for the creation of the fractal. The distribution of the random numbers generated, whether uniform, Gaussian, or otherwise will be discussed later.

The alteration, with the original corner values, varies the fractal dimension. It is assigned before the pixel values are computed. The influence of the alteration value creates the texture or roughness of the fractal. This will be elaborated on in the section describing variables.

After the first four midpoints have been computed a square of eight points is constructed, Figure 1. The center pixel value of the square is computed using

$$pixel = \frac{pixel\,1 + pixel\,2 + pixel\,3 + pixel\,4}{4} + RN \cdot alteration$$

shown in Figure 2. An example of the procedure outlines the computations on the first level of a 513 x 513 pixel fractal. Pixels (1,1), (513,1), (1,513), and (513,513) have been pre-assigned. These values are used to determine the midpoint pixel values:

$$pxl(1,257) = \frac{pxl(1,1)+pxl(1,513)}{2} + RN \cdot alteration$$

$$pxl(257,513) = \frac{pxl(1,513)+pxl(513,513)}{2} + RN \cdot alteration$$

$$pxl(513,257) = \frac{pxl(513,1)+pxl(513,513)}{2} + RN \cdot alteration$$

$$pxl(257,1) = \frac{pxl(1,1)+pxl(513,1)}{2} + RN \cdot alteration\ .$$

The center point is computed with

$$pxl(257,257) = \frac{pxl(1,257)+pxl(257,513)+pxl(513,257)+pxl(257,1)}{4} + RN \cdot alteration$$

As an example, if the four initial corner values, (1,1), (1,513), (513,1), and (513,513), are assigned to be 64, a uniform distribution is utilized, and the alteration is set at 25 the following values would be derived:

$$pxl(1,257) = 84 = \frac{64+64}{2} + (0.8 \cdot 25)$$

$$pxl(257,513) = 55 = \frac{64+64}{2} + (-0.35 \cdot 25)$$

$$pxl(513,257) = 62 = \frac{64+64}{2} + (-0.1 \cdot 25)$$

$$pxl(257,1) = 82 = \frac{64+64}{2} + (0.7 \cdot 25)$$

$$pxl(257,257) = 84 = \frac{84+73+67+82}{4} + (0.3 \cdot 25)$$

The size of the fractal, in pixels , must be determined prior to the computation. This algorithm creates square images that require a limited number of levels to create 2 to the Nth power + 1 pixels per side. The number of levels required for a specific size fractal is defined by, for (N+1) levels, $size = ((2^N + 1)pixels)^2$.

In practice this means that fractals can be created of 3 x 3, 5 x 5, 9 x 9, 17 x 17, 33 x 33, etc. pixels. The fractals created in this work were 513 x 513 because of the display screen size of 512 x 512. The computer code for this algorithm, in FORTRAN 77, has been included in Appendix B.

B. VARIABLES

The algorithm is simple but differences can be produced by altering the values of the variables: the starting corner values, RN, and alteration. Variables other than those specifically mentioned are held constant. For this section the random number distribution is uniform, corner values are 128, and the alteration is, from step one to step nine, 64, 32, 16, 8, 4, 2, 1, 1, 1.

1. Corner Values

The corner values influence the mean of the fractal in terms of its grey level. Values can be assigned separately for each corner pixel but all corner pixel values must be between 0 and 255. Figures 3 and 4 show the effects of a change in the corner value from 16, 32, 64, and 128 in

Figure 3 to 64, 64, 64, and 64 in Figure 4. The seed for both fractals is one. Figures 3a and 4a show the monochrome image, Figures 3b and 4b show their wire mesh representations.

The beginning corner values will influence the grey levels in the fractal to remain at a corresponding level. If the random number has an equal chance of being positive or negative the value of each change, which is the alteration multiplied by RN, has an equal chance of increasing or decreasing the value of the computed pixel from the average of the values from which it is derived. This is the essential nature of a random walk. The direction or distance taken does not rely on previous movements.

Extremely low or high values will be artificially controlled through clipping at 0 and 255. Eighteen fractals were created with a constant seed of 50, uniform distribution, and a constant alteration of 64. The corner values were different for each fractal, 5, 50, 75, 100, 125, 150, 175, 200, 225, 250, but the values were the same for all four corners of each fractal.

The means, Figure 5, and the standard deviations, Figure 6, show the influence that the corner values have on the mean value of the fractal. In Figure 5 the low and high values show the effect of clipping the pixel values at 0 and 255 on the mean. The inability of pixel values to exceed

255 cause the mean to be less than would be expected with a corresponding corner value. The same is true at the low end of the grey scale with the mean being slightly higher than the corner value. Figure 6, the standard deviations for varying corner values, show the expected decrease at the high and low ends again caused by clipping.

2. Random Number Generation

There are two aspects to RN, the random number: the seed and the distribution. Each individual seed provides a singular construct for the fractal. The distribution is the type of frequency distribution of the random numbers provided by the computer: binomial, Gaussian, or other. The seed is the number given to the computer by the operator to begin generation of random numbers. Use of the internal clock of the computer as a seed generator allows pseudorandom assignment of the seed value. All non-zero integers can also be used as the seed.

The storage of the seed value allows the operator to recreate the same fractal. This ability results in storage savings for fractal images. A 512 x 512 image would normally use 262,144 bytes of storage if each pixel is represented by one byte. By storing the corner value, the seed, and the alteration value the storage space decreases to as little as three bytes along with the storage space for the program. Regeneration time of the fractal is dependent upon the computer. A VAX 11-780 can recreate a fractal in approximately 45 seconds.

The random number generator influences the distribution from -1 to 1 of the alteration. A constant alteration would vary the product of the alteration with the random variable from the negative to positive values according to the distribution. The random numbers for these fractals were provided by the International Mathematics Subroutine Library (IMSL) utility installed on a VAX 11-780.

When a Gaussian distribution is used values of -1 to 1 are returned by the computer. It has a mean of zero with an approximately bell-shaped curve. This would tend to change the mean of two pixels very little for the value of the midpoint pixel since the product of the alteration and the random number concentrates around small values. Figure 7 shows the effect of a Gaussian distribution on the fractal. The mean of the fractal is 18.4 and the standard deviation is 29.1.

In the case of Figure 8 the binomial distribution has a probability value of 0.8 calculated over 20 events for values selected by the operator and coded into the software. IMSL returns real numbers between 0 and 1. This is modified by scaling the real numbers returned by the computer to a range between -1 and 1. The mean of these returned values is biased, not 0. The mean of the fractal created with a binomial distribution is 42.7 with a standard deviation of 26.4. There is a negative bias to both the Gaussian and the binomial images that have a dominant effect on those shapes.

A uniform distribution allows an equal chance at the random walk anywhere between a negative product value and a positive product value. IMSL returns a value between 0 and 1 which is scaled to -1 and 1. The standard deviation of the returned values is $1/\sqrt{12}$. The standard deviation of the rescaled values is $1/\sqrt{3}$.

Figure 9 has all the same input values as Figure 7 and 8 except for a uniform distribution. The uniform distribution created a fractal, Figure 9 with a mean of 43.2 and a standard deviation of 20.6. The seed utilized in Figures 7, 8, and 9 is 56.

3. Alteration

The variable with the closest relationship to the overall appearance is the alteration. This variable affects the texture of the fractal, giving, visually, the impression of rolling, rough, or mountainous terrain. The fractals created as examples in this section all use the same seed of 111. Fractals created with the same seed usually have some feature in common that indicates their common source. There are a variety of ways to implement the alteration in this algorithm. The roughness of the fractal is dependent upon the relation of the alteration size to the step in the fractal creation process. In a 512 x 512 pixel fractal there are ten stages. The first stage creates new pixels that are 256 pixels removed, vertically and horizontally, from the original corner pixels. As the algorithm moves through the creation process, the physical distance between newly created pixel values and source pixels becomes closer and closer.

The mean of the source pixels is the basis for the value of the new pixel. Added to that value is RN, with a minimum of -1 and a maximum of 1, multiplied by the alteration. If the alteration remains constant throughout the algorithm and is a large number, compared to the range of possible values for a pixel the final fractal will be rough, as in Figure 10, where the alteration is 64.

This occurs because in the last few steps of the fractal creation, the spacing between new pixels and the creators is small, but the new pixel value, influenced by the alteration, can make large values changes from adjacent pixels. The fractal will be rough with a high frequency of value changes between fractals. If the alteration is small, the fractal will be smooth with smaller value changes between pixels possible, Figure 11. The distribution used here is uniform.

The alteration can also be changed at each step of the process. Figure 12 shows the alteration increasing as the distance between new pixels and creator pixels decrease. A rough texture results because the jump in pixel value is high where the physical distance between pixels is low. Figure 13 shows the opposite case. The alteration decreases as the distance increases. A smooth fractal image results.

Different alteration sets are applicable to different bottom types. A set of alterations that decrease and is clipped at 0, refer to Figure 7, is representative of coral or rocks surrounded by mud or drifted sand. The same alteration values, without clipping, would represent a completely rocky region. However a different set of alteration values is representative of a rough bottom covered with gravel. Figure 10, with a constant alteration value of 64, is an example of what could be a gravel bottom. The fractals can be scaled to provide a rough texture with a small range of values. A fractal can also represent an area of one square foot or one square mile. Figure 9 as a description of an area of one square foot is considerably rougher than if it described a much larger area.

C. JUSTIFICATION AND VALIDATION

Validation of data simulated by a fractal algorithm would involve three tests comparing real to simulated data. Real data would be represented by digitized input from texture images. Since the purpose was to simulate the ocean bottom a selection of textures typical of the ocean bottom was used. The grey level of each pixel would represent height of the topographical features of the image.

First, since the data must appear visually like the data it purports to simulate, subjects must compare simulated to real data in a visual comparison. Second, a statistical analysis must evaluate real and simulated data. Third, a test will evaluate the fractal dimension of simulated data and compare it to the fractal dimension of real data that the algorithm has attempted to simulate.

1. Visual Inspection

The first test was carried out by showing two subjects a series of images. They were asked to evaluate which images were taken from digitized images of texture and which had been simulated using the fractal algorithm. This test evaluated texture, grey levels, and the size of the features in the image. The subjects were not given directions but were asked which images appeared to be examples of topography and which appeared to be simulations.

The two judges presently work in the area of signal and image processing. Their work involves sonar and laser data. One evaluator is an electronic engineer specializing in signal and image processing and the other is a mathematician, specifically in the area of probability. Both have been involved for several years with detection and classification of objects in images.

The images of the real data are presented in Figures 14 - 17. Figure 14a and b are sand and the mesh representation of sand. Figure 15a is an image of a rough wall, 15b is the mesh representation. Although it is highly unlikely that a wall would be found on the ocean bottom the texture is not unlike some corals, lava flows, and pitted rock formations. The next image set, Figure 16, is also sand but has a different texture from Figure 14. The last image, Figure 17a and b, is gravel and its mesh representation. The images are digitized from the University of Southern California texture image set.

The simulated images are shown in Figures 18 - 21. The mesh representations were viewed by the two judges and all were deemed to be topographical. Figure 21b, which is noticeably different from the other representations was still felt to be topographically representative. When asked to compare images that were statistically similar there were two sets accepted and two rejected. The agreement between Figures 14 and 18 and Figures 16 and 20 were judged acceptable. However Figures 15 and 19 and Figures 17 and 21, in both a) and b) of each figure, were not judged to be similar in appearance.

2. Statistical Analysis

A statistical analysis was run on all images, real and simulated, providing pixel grey level distribution, mean, and standard deviation. It was expected that a realistic (acceptable) simulation would have statistics similar to the type of data that it what intending to simulate. The mean and standard deviation was 155 and 32.06

for Figure 14, 184 and 36.28 for Figure 15, 212 and 31.52 for Figure 16, and 199 and 35.72 for Figure 17.

The statistics of the simulated images was controlled through the variables. The real images had means of 155, 184, 212, and 199. The simulated images were created with corner values of 155 for Figure 18, 185 for Figure 19, 210 for Figure 20, and 200 for Figure 21. The corner values resulted in means of 150, 171, 225, and 193. The standard deviations for those images were 35.8, 36.0, 28.0, and 35.5.

Figure 18 had an alteration set of 5 to 45 in increments of 5. This resulted in a rough texture similar to the sand in Figure 14. The mean of Figure 14, at 155, is close to Figure 18 at 150. The standard deviations of 32.06 and 35.8 are similar.

Figure 19 has a standard deviation of 36.0 compared to 36.28 for Figure 15. The alteration set for Figure 19 was a constant value of 25. A constant value for the alteration creates a standard deviation in the resulting fractal close to the value of the alteration. The mean of 171 approximates that of Figure 15, 184, by use of the corner value of 185.

The next set of images, Figures 16 and 20, have similar means, 212 and 225, and standard deviations, 31.52 and 28.0. The alteration set used was 1, 1, 1, 2, 4, 8, 16, 32, 64. This alteration set produces a rough textured image. The image it is meant to simulate is the second image of sand.

The last set of images, Figures 17 and 21, are statistically similar with means of 199 and 193 and standard deviations of 35.72 and 35.5. The alteration set was 64, 64, 64, 32, 16, 8, 4, 2, 1. This set produces a smoothly textured image.

3. Fractal Dimension

Although several methods are applicable for measuring fractal dimension the method used here is described by Dubuc et al(Dubuc, et al. 1989, pp. 113-127). The fractal dimension of a surface can be measured as a non-integer between two and three. Dubuc et al. have developed a method for estimating the fractal dimension of a surface that is described as robust and can be used on digitized data. This variation method is suitable because the use of digitized data can be used as input without a loss of accuracy that is found when using classical algorithms. The Fortran code for this method to determine the dimension of an image is presented in Appendix B. It is known as the difference statistics algorithm where the fractal dimension, D, is

$$D = \lim_{\varepsilon \to 0}\left(3 - \frac{\log A(\Delta_x, \Delta_y)}{\log \varepsilon}\right).$$

A is the mean pixel value at position x, y and is the difference in value from the average and the actual value.

Figures 14 through 17 were all similar. The image of sand, Figure 14, was 2.19 calculated over subsets of 64 by 64 pixels. Figure 15, the rough wall, under the same circumstances had a dimension calculated of 2.15. Figures 16 and 17 correspondingly had dimensions of 2.21 and 2.16. Although each image looks different in the a) portion of each figure, one from the other, their appearance in the mesh representations support the similarity of their dimension values.

III. CONCLUSION AND RECOMMENDATIONS

The results of the three tests were encouraging. The statistical analysis of the real images versus the fractal images showed the statistics of the fractal images could be manipulated during their creation. The mean of an image can be controlled through the choice of corner values. The standard deviation of a fractal, for this algorithm, can be controlled through the use of the alteration set. A constant alteration will result in a standard deviation close to the alteration value. An increasing alteration set will result in a highly textured surface. A decreasing alteration set will result in a smooth, convoluted surface.

The dimension determination showed consistent values for the fractals used to simulate the images in Figures 14 - 17. However the dimension of the fractal can also be controlled to some extent through the choice of the alteration set and the corner values. In creating the fractals, test cases showed that a higher dimension could be obtained with a corner value of 25 and a decreasing alteration set. The fractal dimension that was created was adequate for the purposes of this study.

The visual inspection of the simulated images was the least successful of the three tests. However a comparison between an image and its mesh representation indicates that the image itself is not the best means of comparison. The human eye is incapable of distinguishing between 256 grey levels and much information in the image is disregarded. The mesh representation gives a better indication of height and space. The nature of a fractal is such that it is size invariant and although an image such as Figure 21 is not a good representation of gravel at one measured size it may be appropriate at another. For example if Figure 21b were considered to represent a square meter of area it would not appropriately represent gravel, but at an area of a square decimeter it may be acceptable. The judges accepted all the fractals to be representative of real data even if they were not visually like the specific images that they intended to represent.

As a recommendation, more work can be done in several areas. First, the manipulation of the fractal dimension can be studied further, perhaps with comparisons of different methods of determining the dimension. Second, more comparisons of different types of textures could be performed. Third, natural features on the ocean floor, such as sand ripples, could be added.

The algorithm as it now stands could be added to sonar models to provide a better scenario for the modeling of objects on the ocean floor. Run time is minimal and storage space of variables for a specific fractal is as little as three bytes. A limited number of bottom textures is available through this research with the added enhancement of size invariance which would allow multiple uses of a few chosen fractals through size redefinition.

APPENDIX A

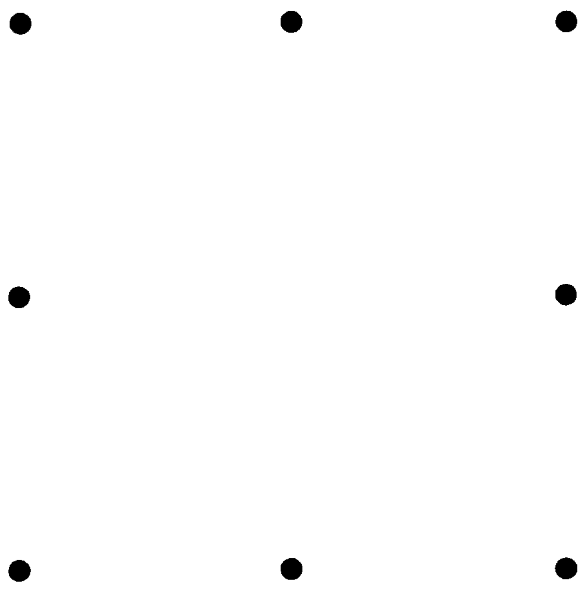

Figure I

Figure 1. Step 1 in Fractal Creation

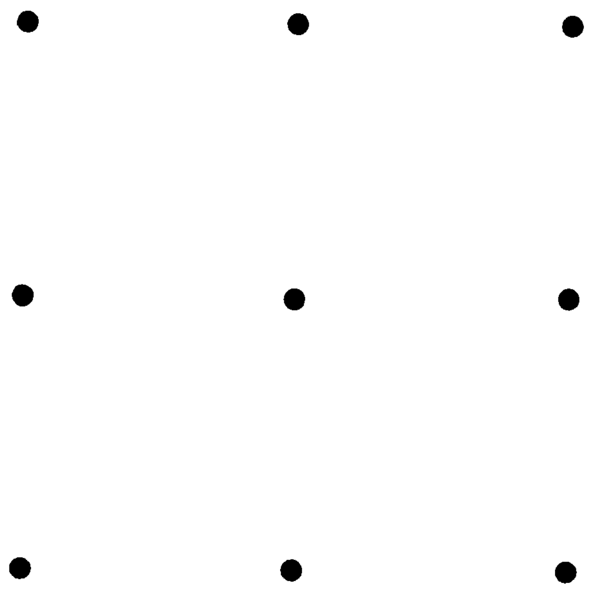

Figure II

Figure 2. Step 2 in Fractal Creation

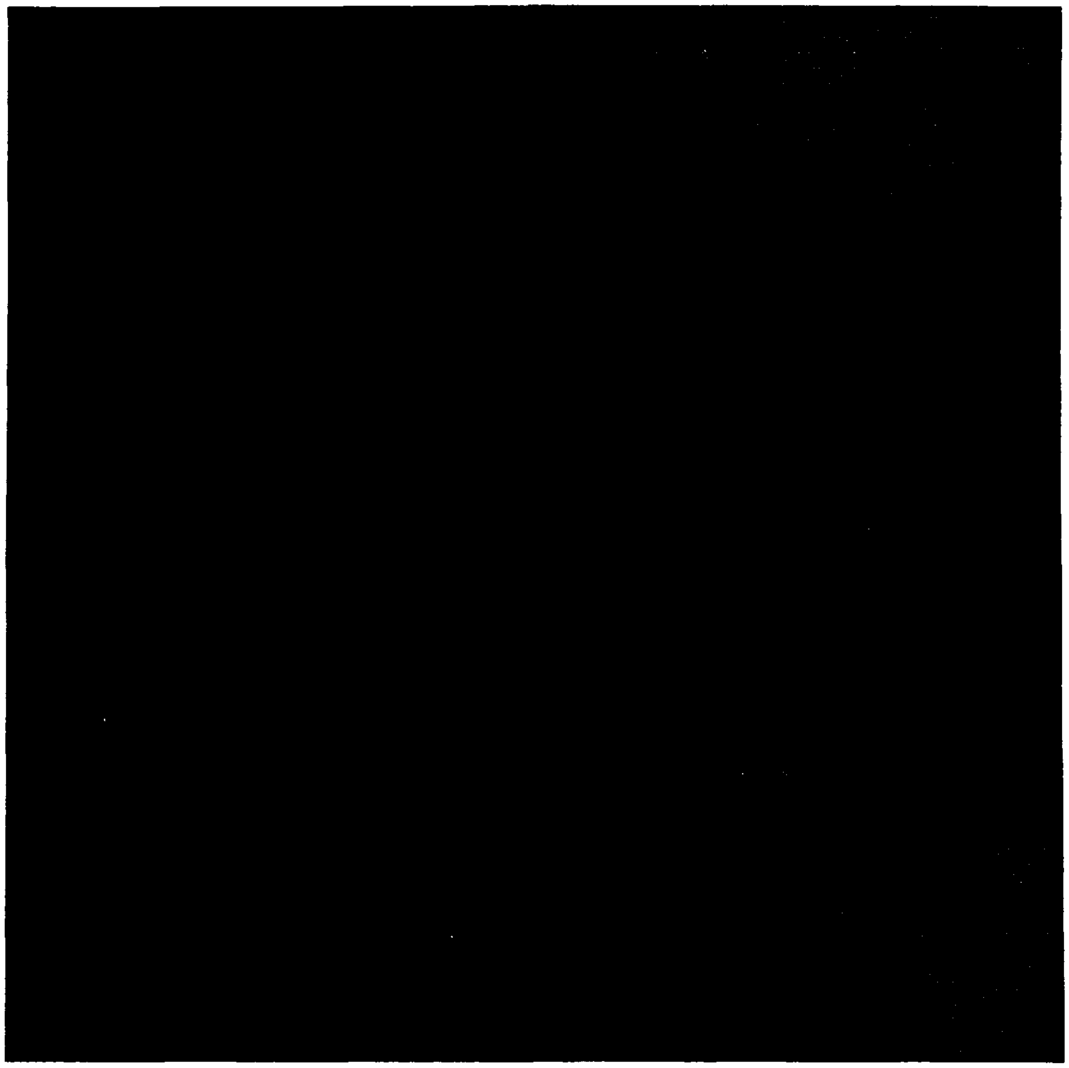

Figure 3A. Corner Values 16, 32, 64, 128

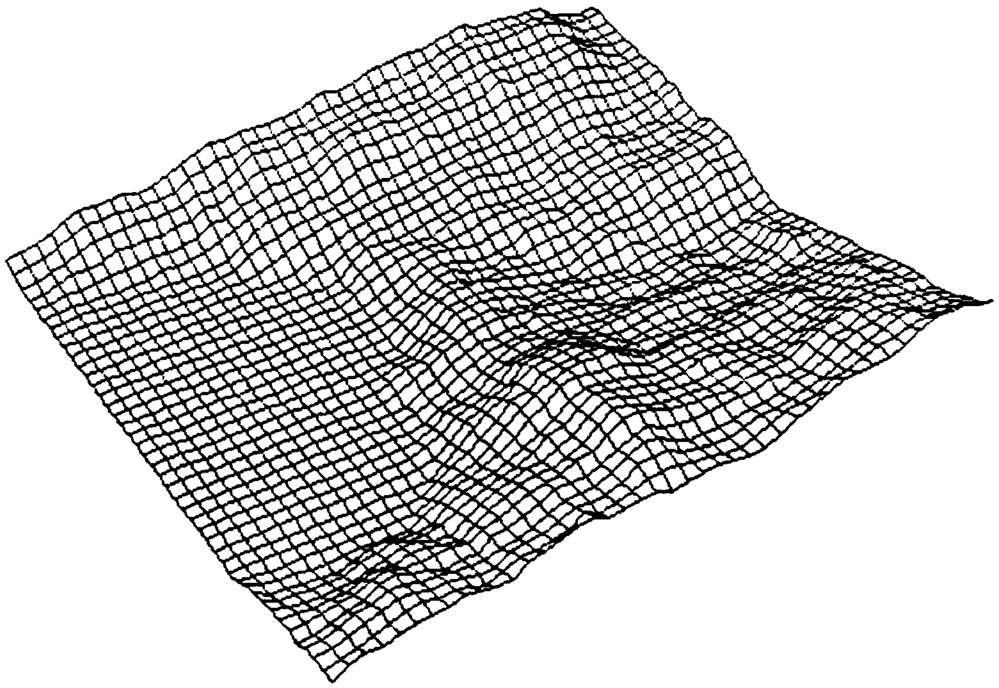

Figure 3B. Mesh Representation - Corner Values 16, 32, 64, 128

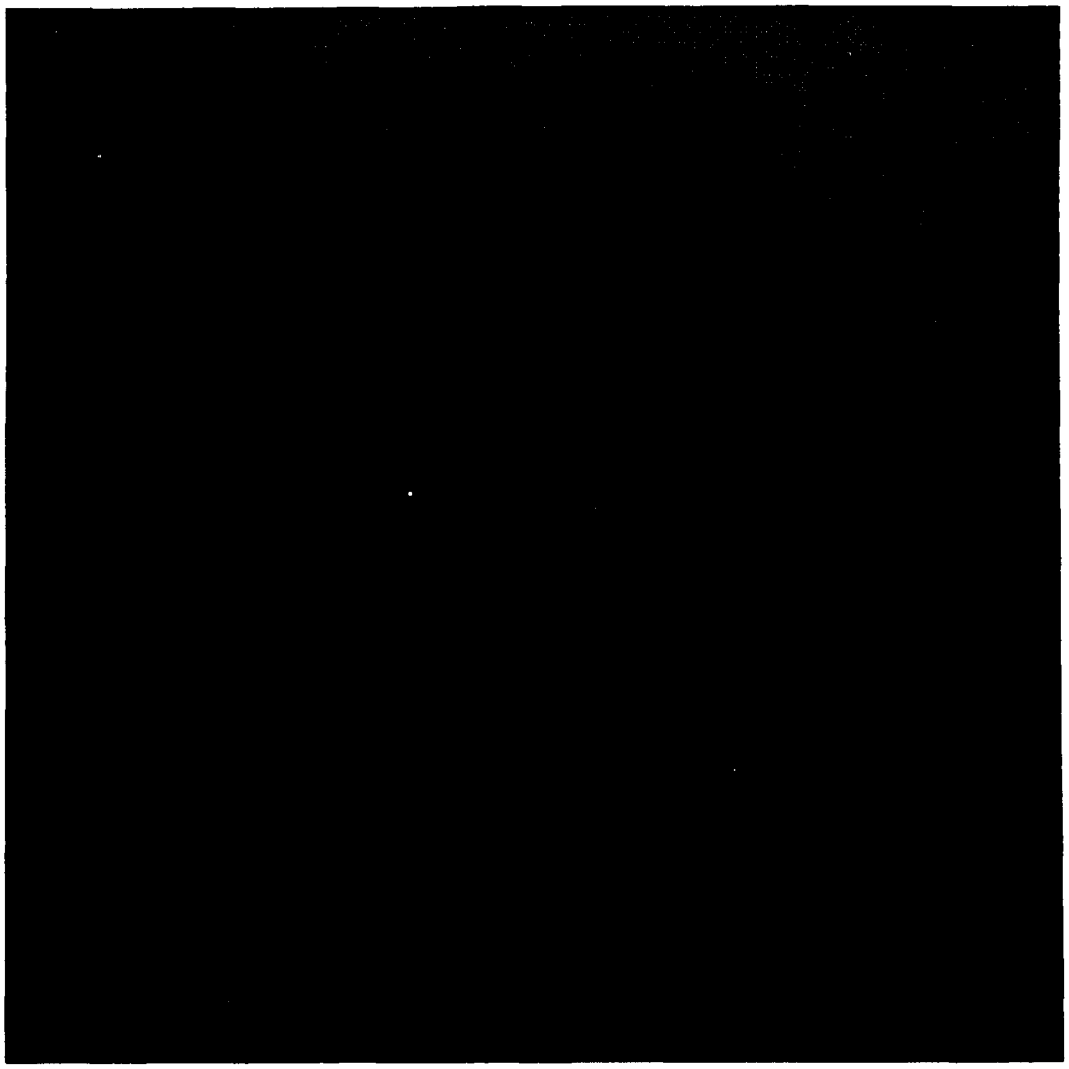

Figure 4A. Corner Values 64, 64, 64, 64

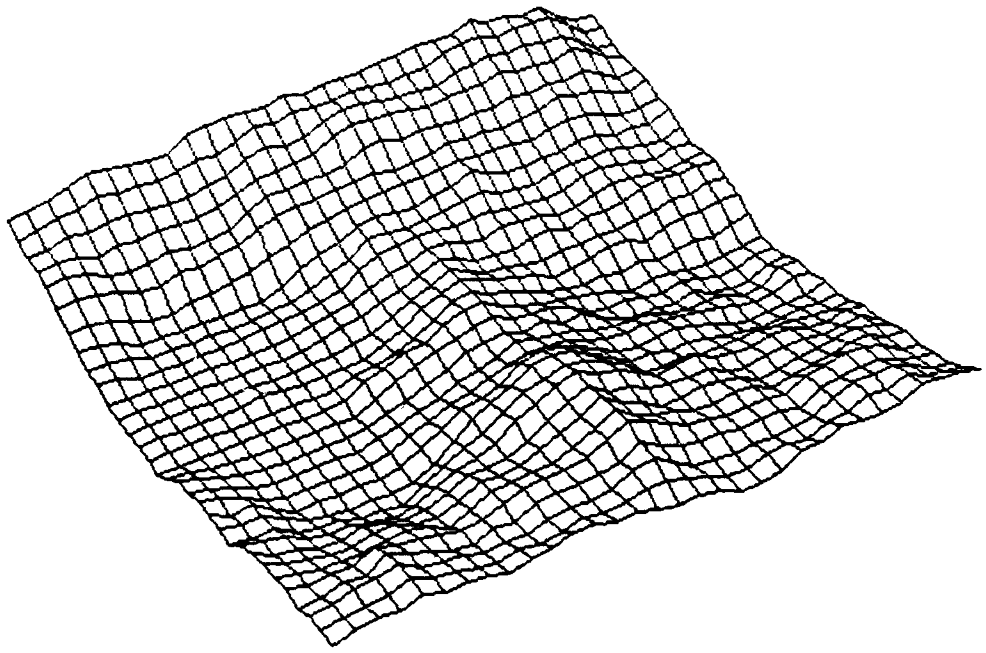

Figure 4B. Mesh Representation - Corner Values 64, 64, 64, 64

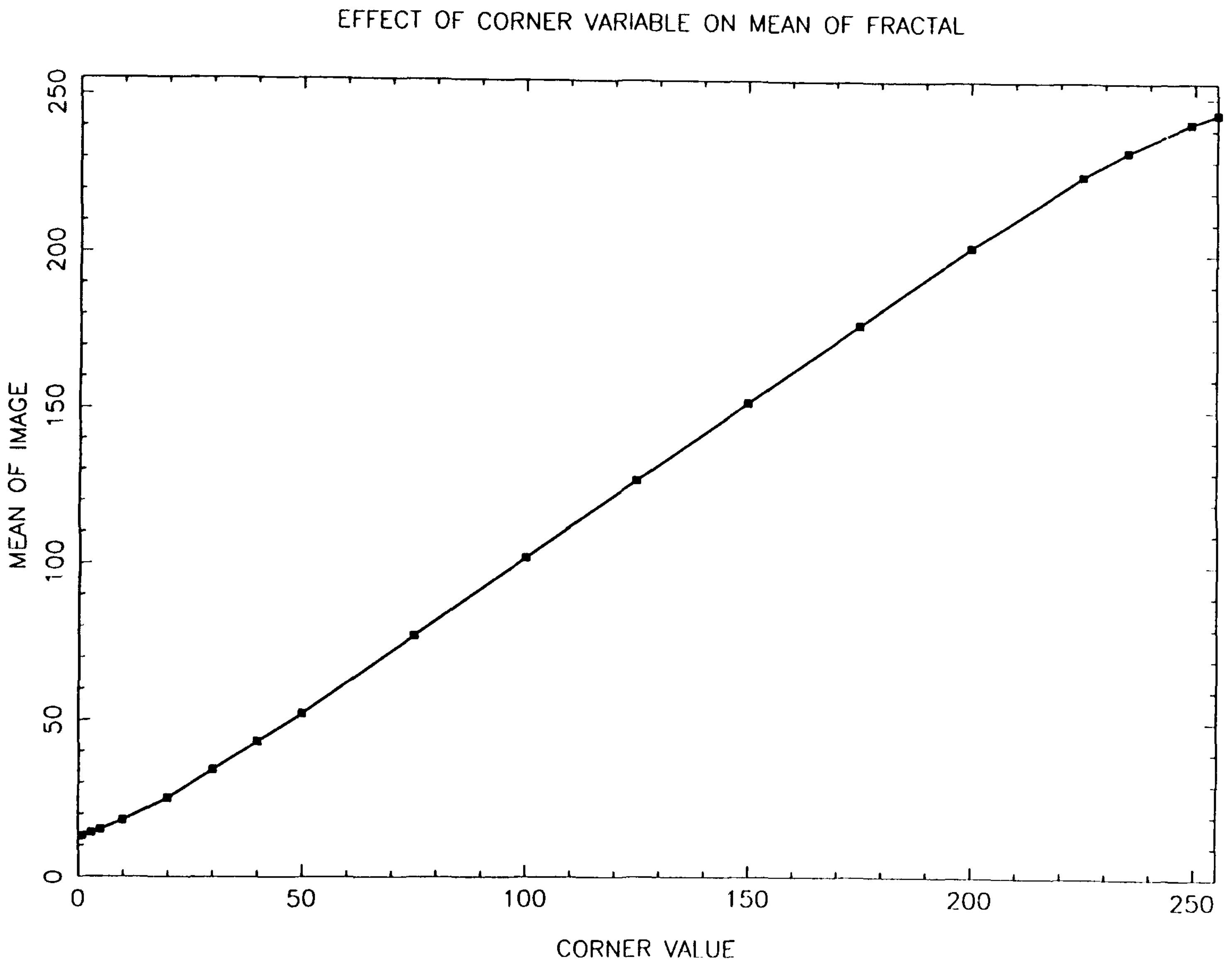

Figure 5. Effect of Corner Variable on Mean of Fractal

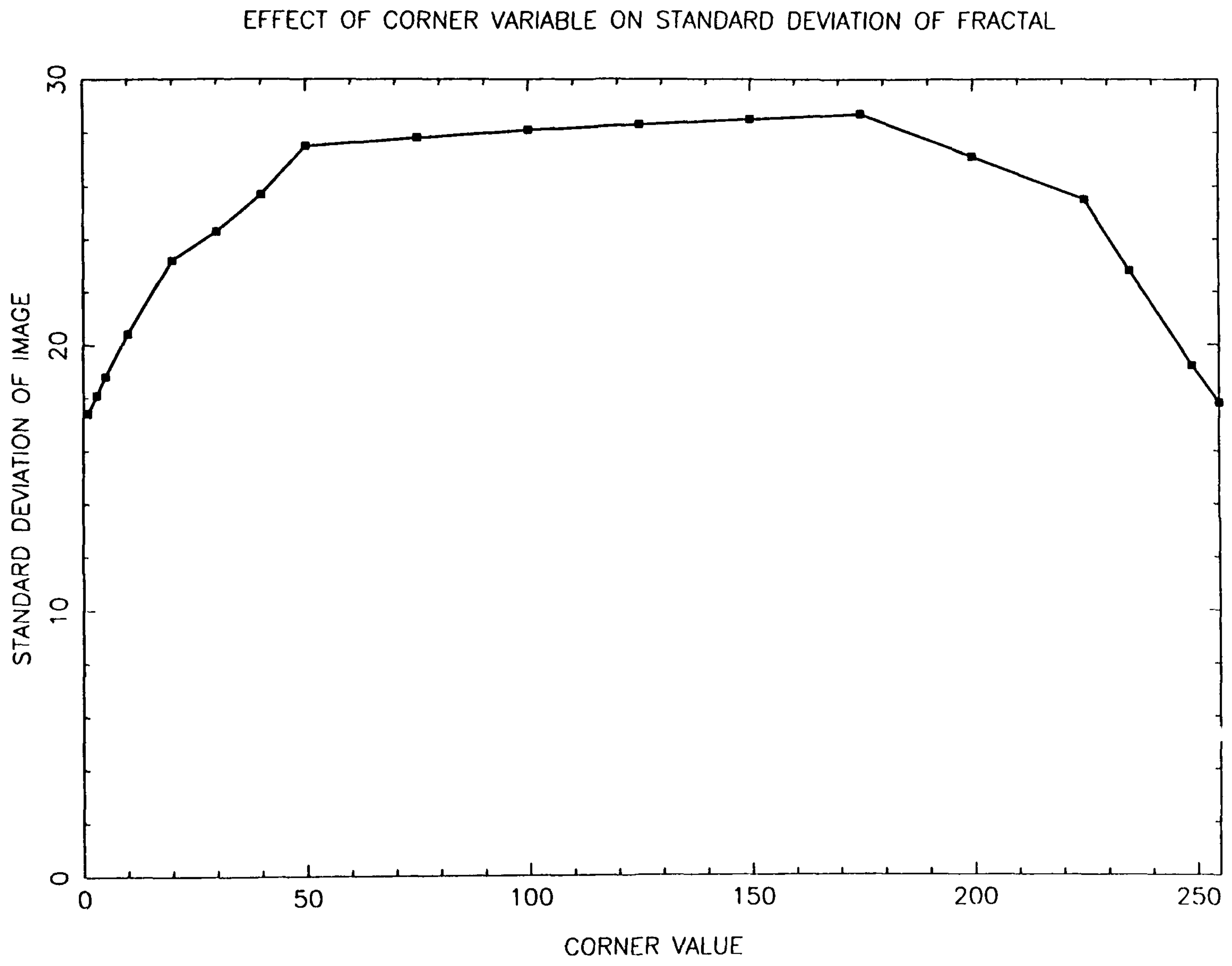

Figure 6. Effect of Corner Variable on Standard Deviation of Fractal

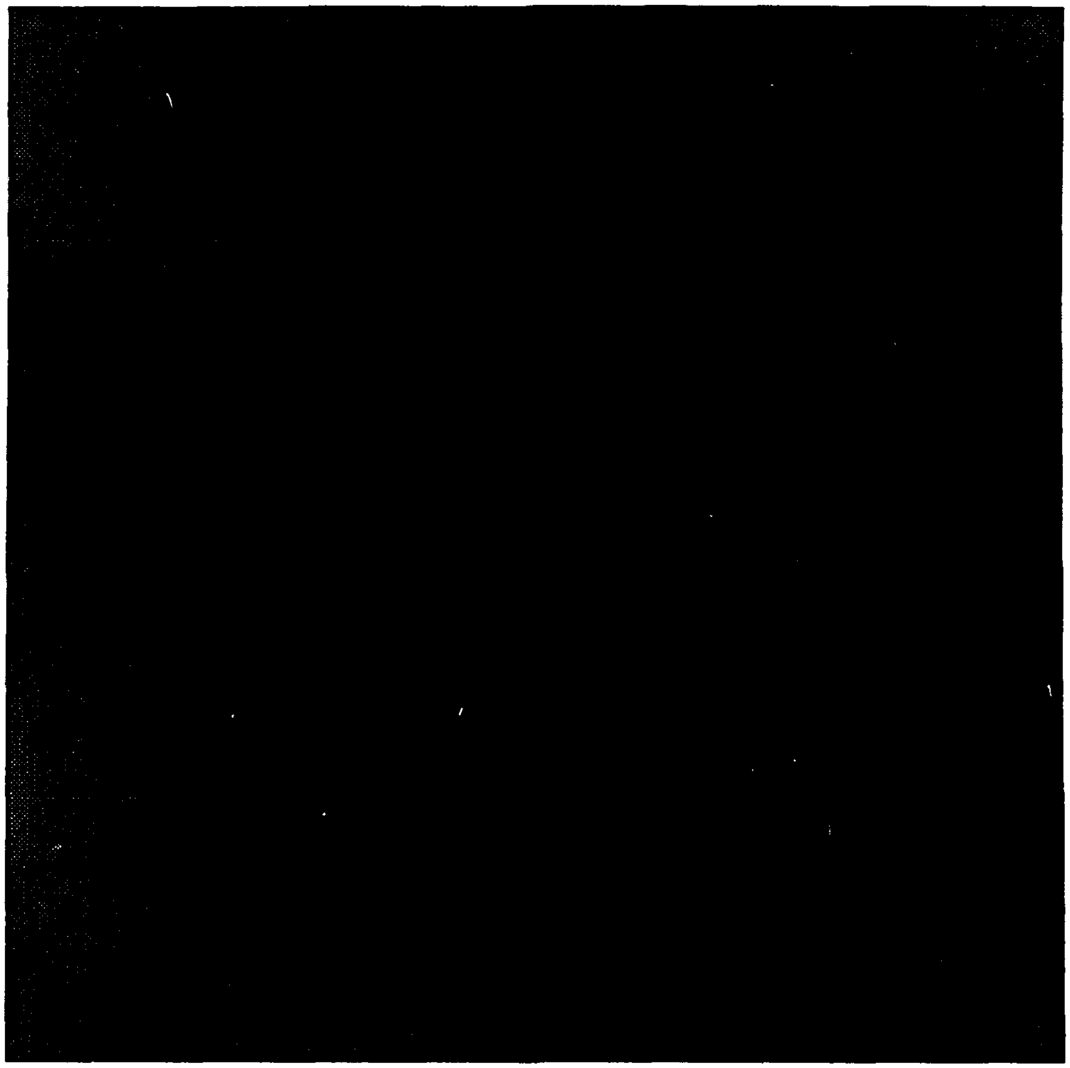

Figure 7A. Gaussian Distribution

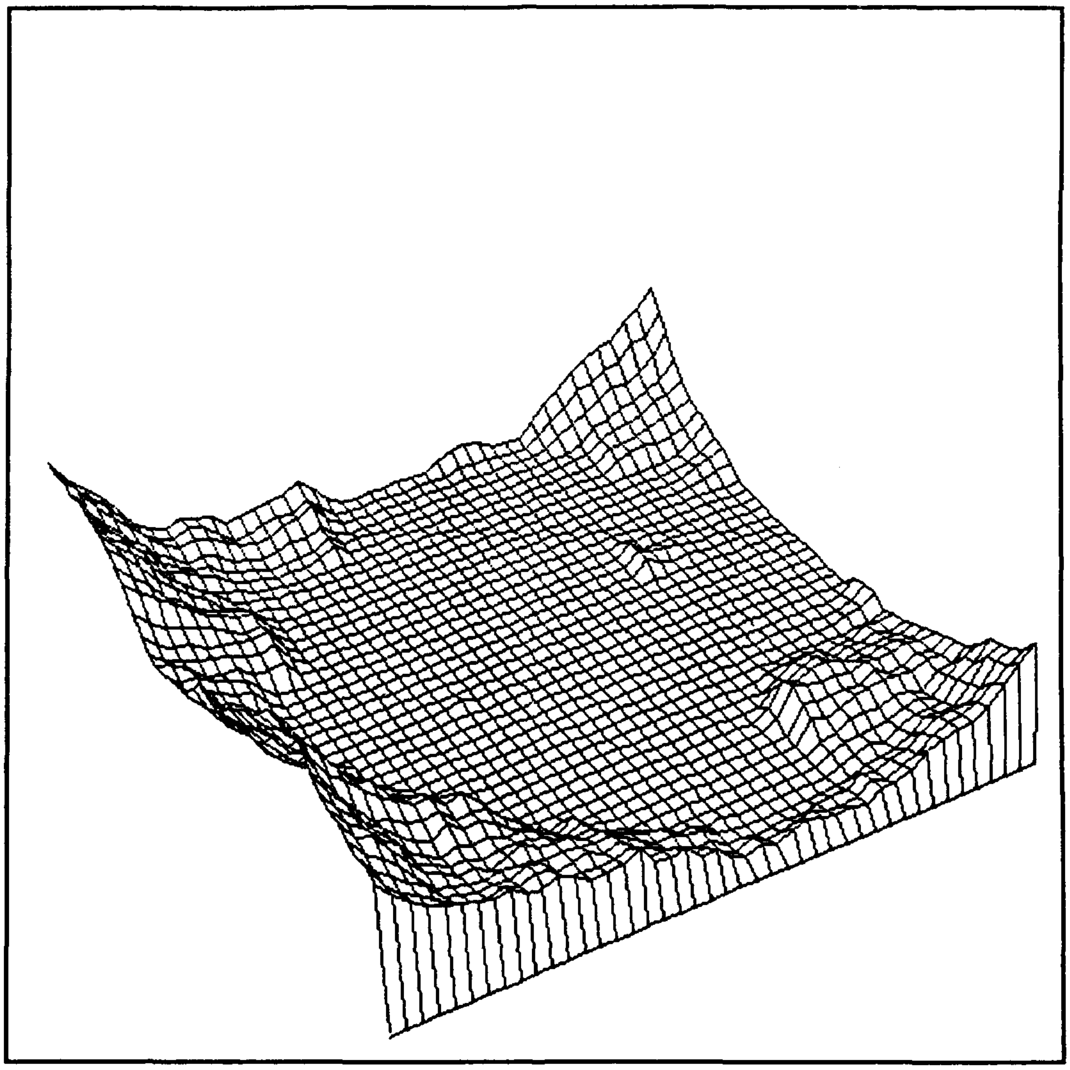

Figure 7B. Mesh Representation - Gaussian Distribution

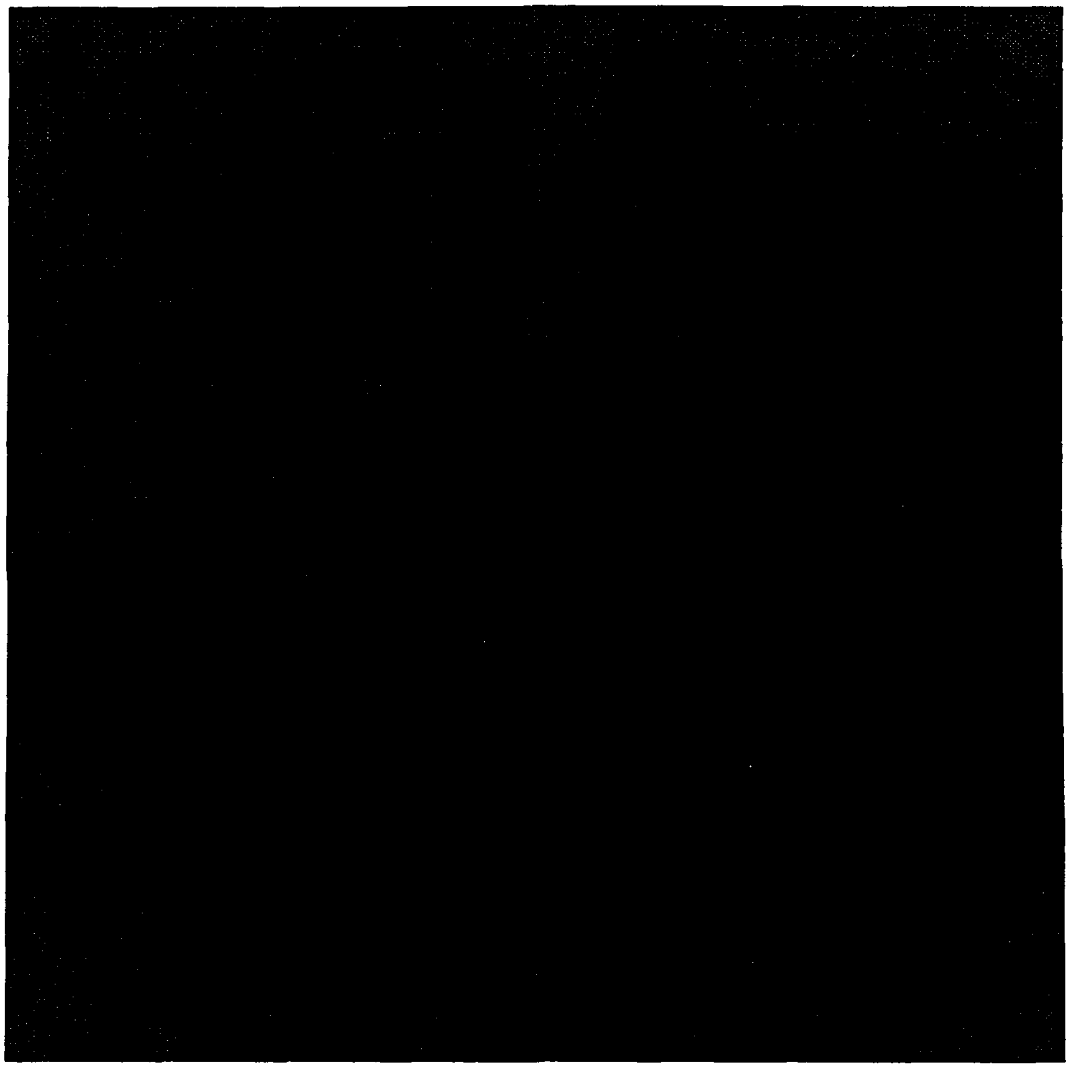

Figure 8A. Binomial Distribution

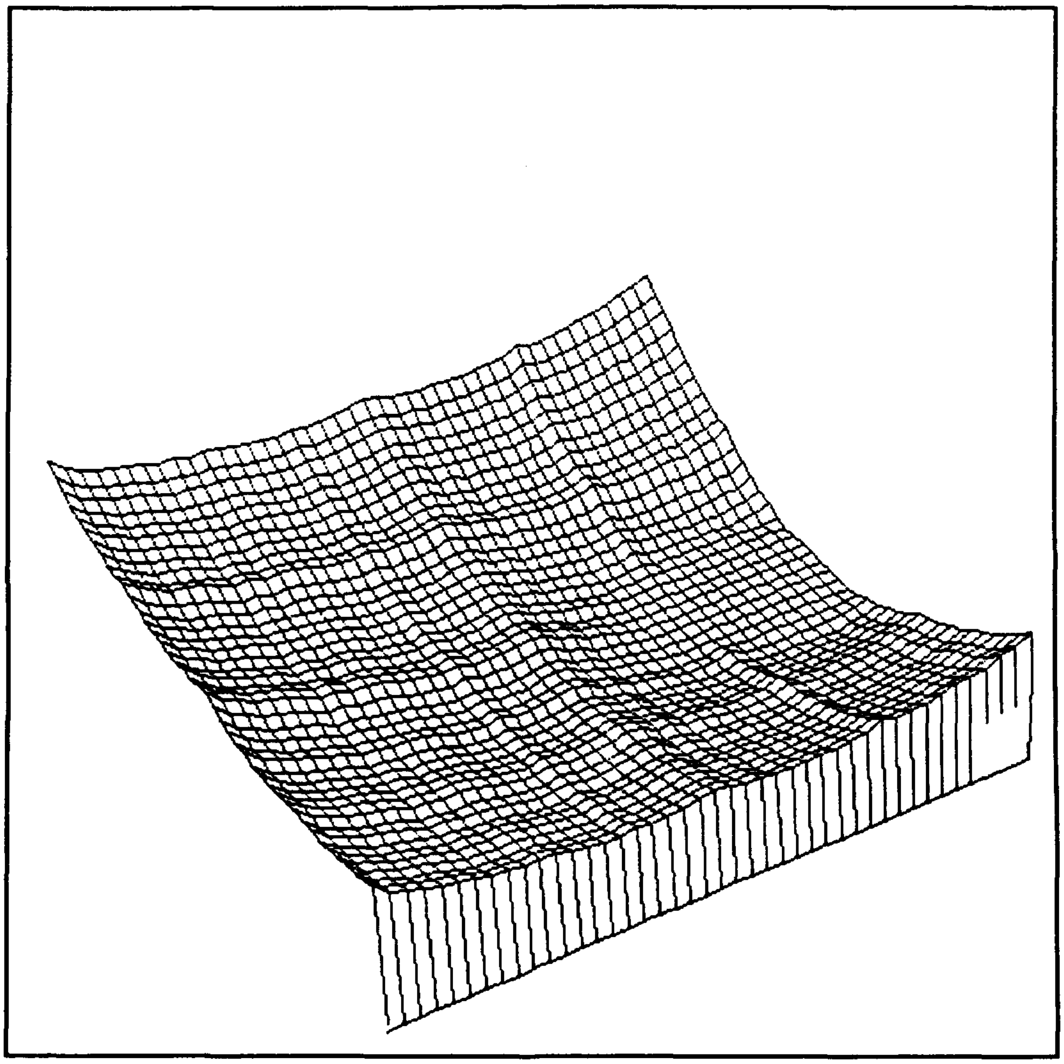

Figure 8B. Mesh Representation - Binomial Distribution

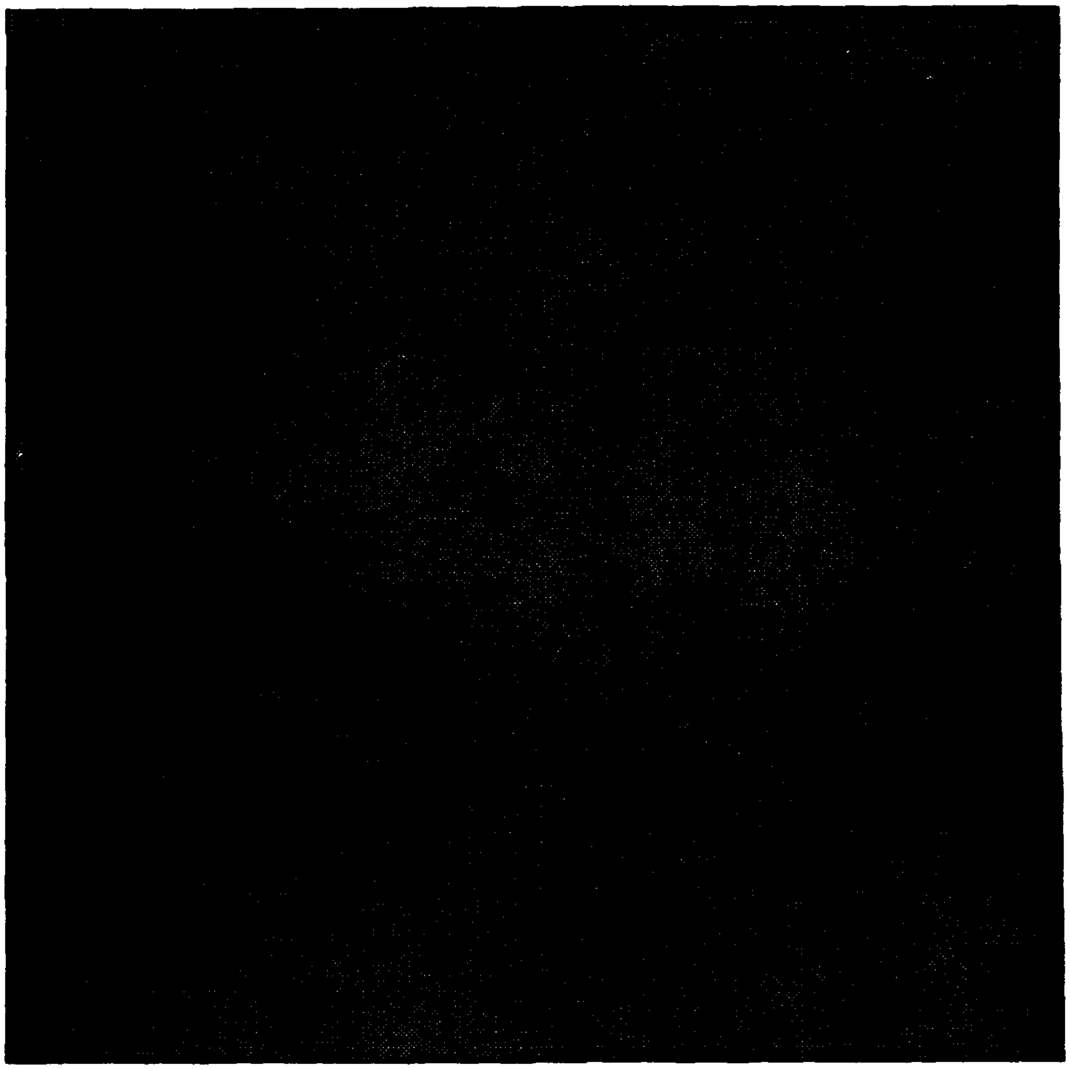

Figure 9A. Uniform Distribution

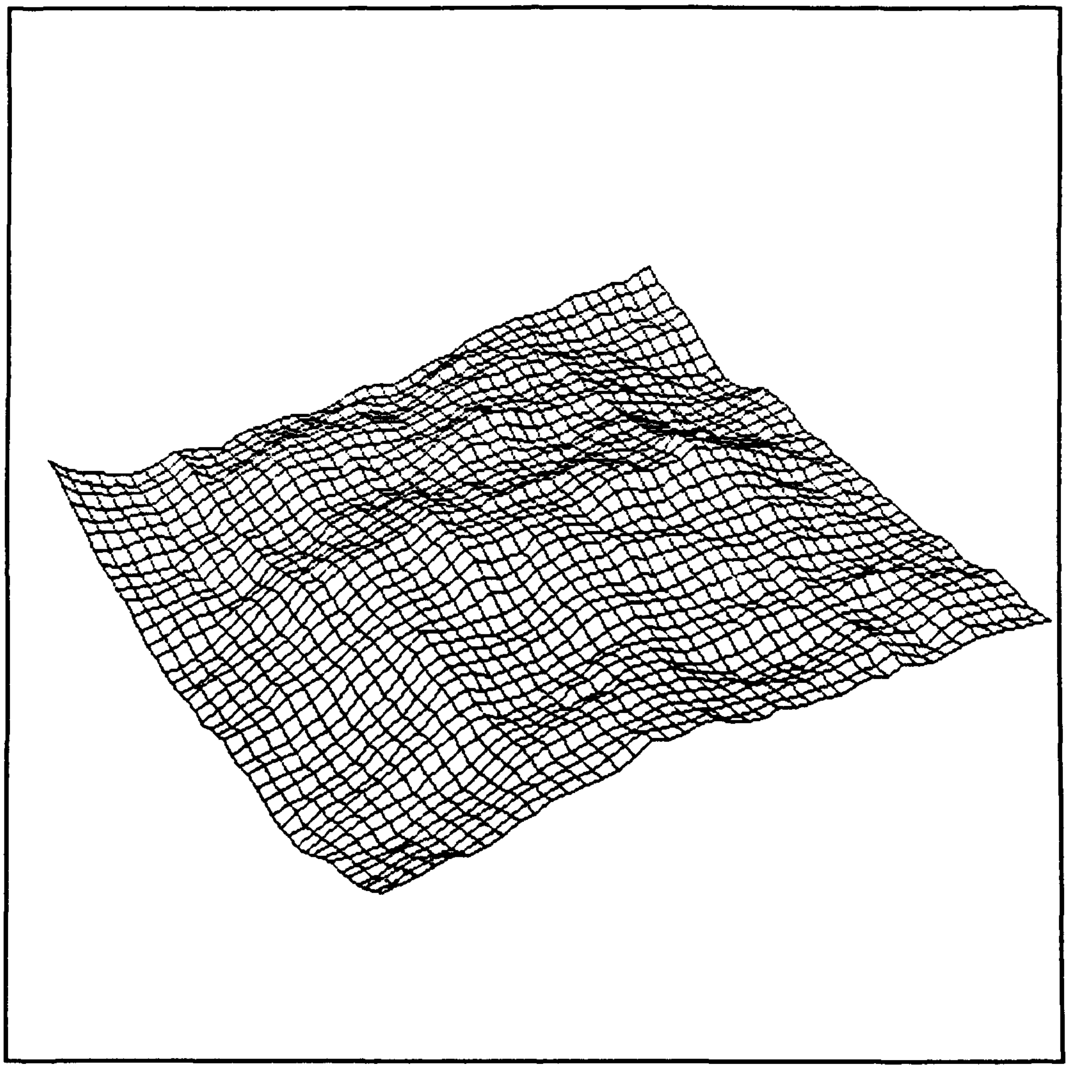

Figure 9B. Mesh Representation - Uniform Distribution

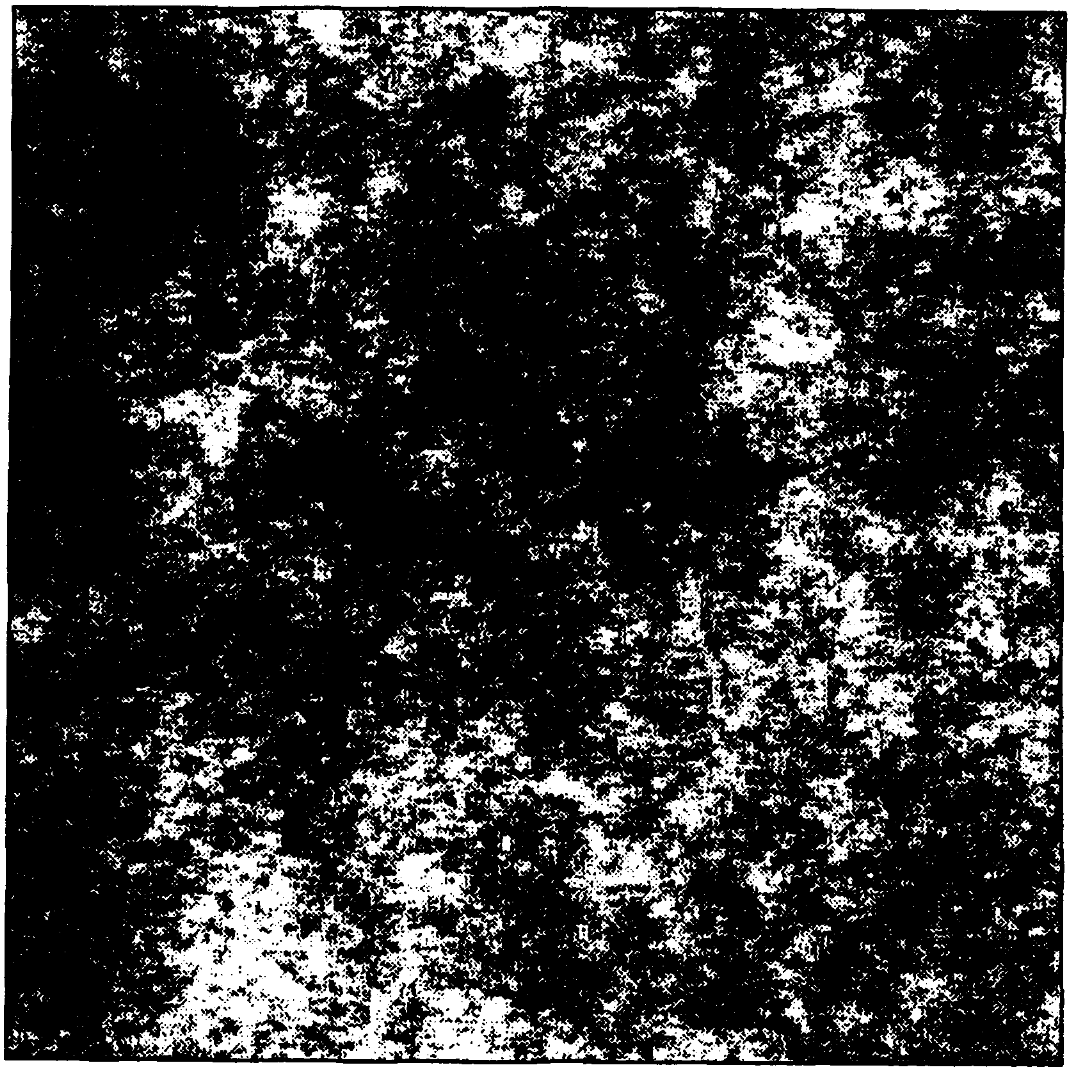

Figure 10A. Alteration - Constant of 64

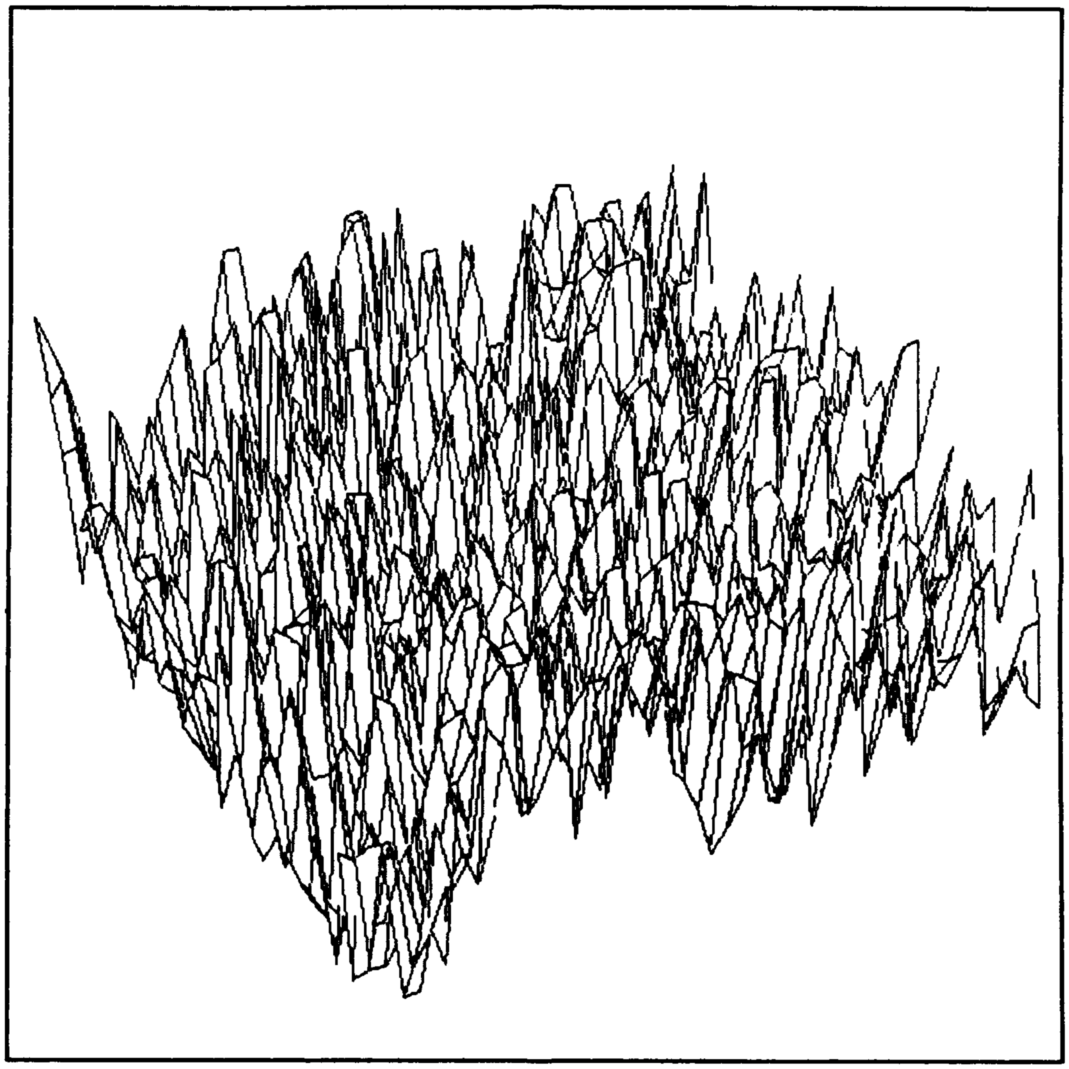

Figure 10B. Mesh Representation - Constant of 64

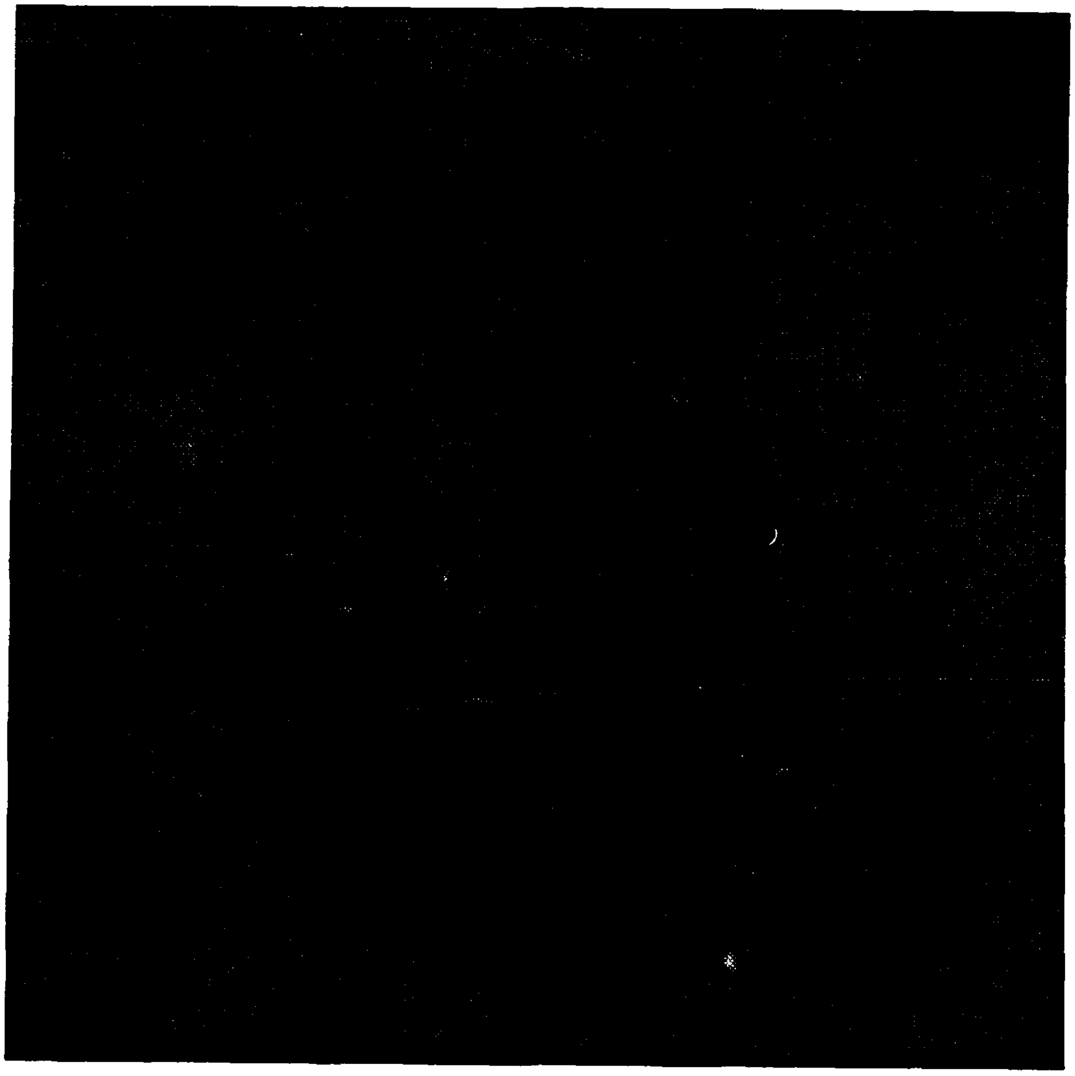

Figure 11A. Alteration - Constant of 5

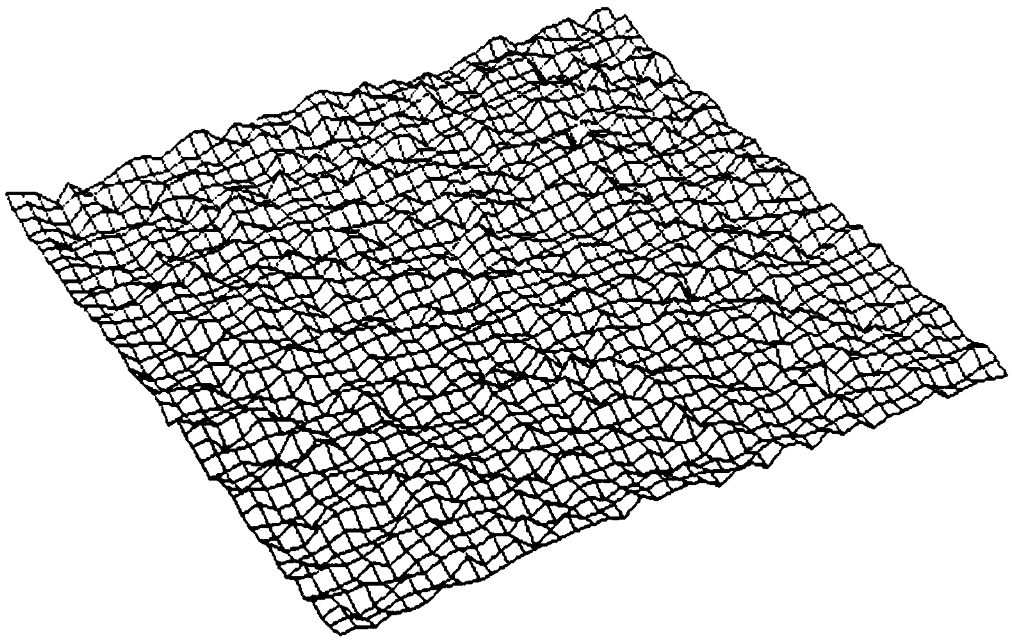

Figure 11B. Mesh Representation - Constant of 5

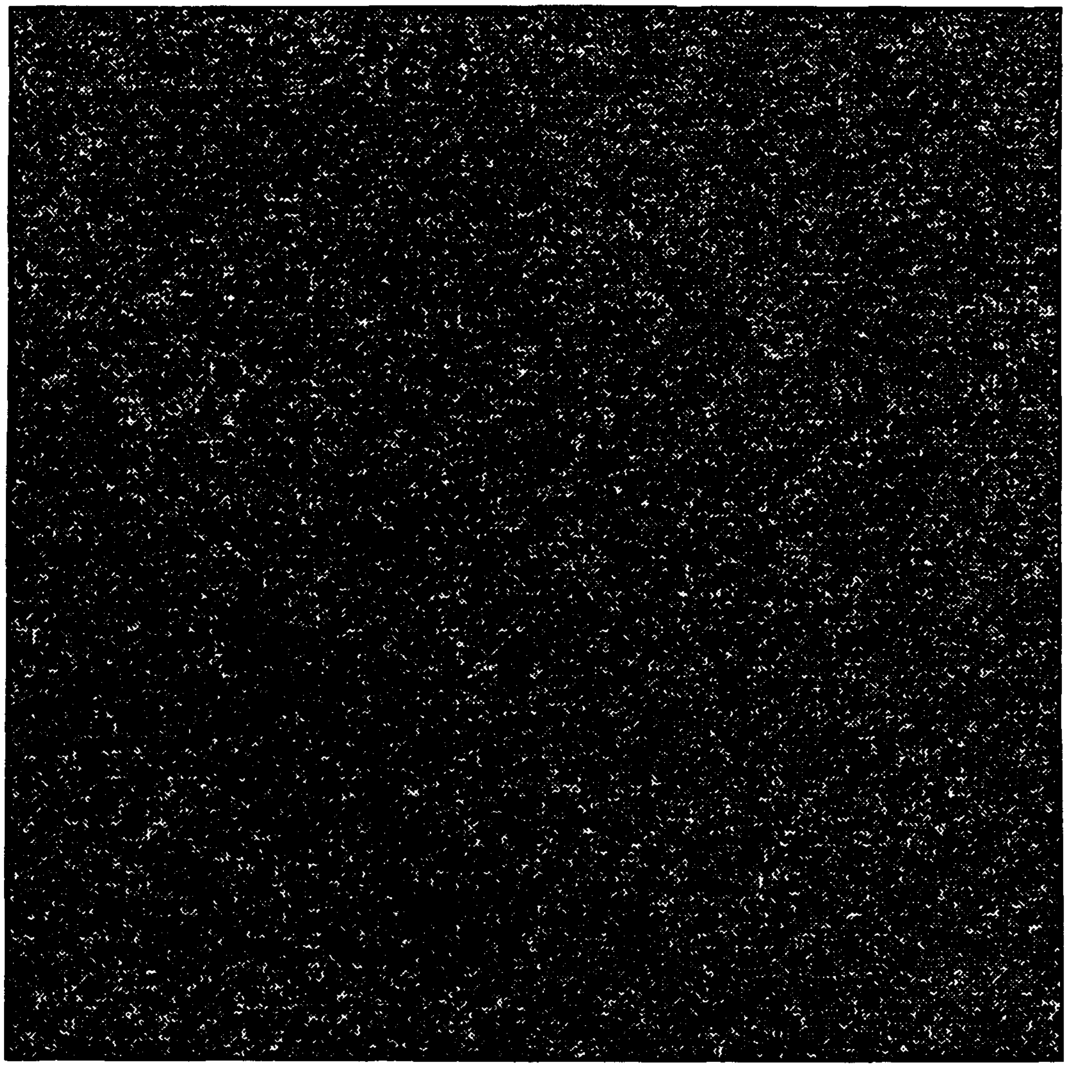

Figure 12A. Alteration - Increasing

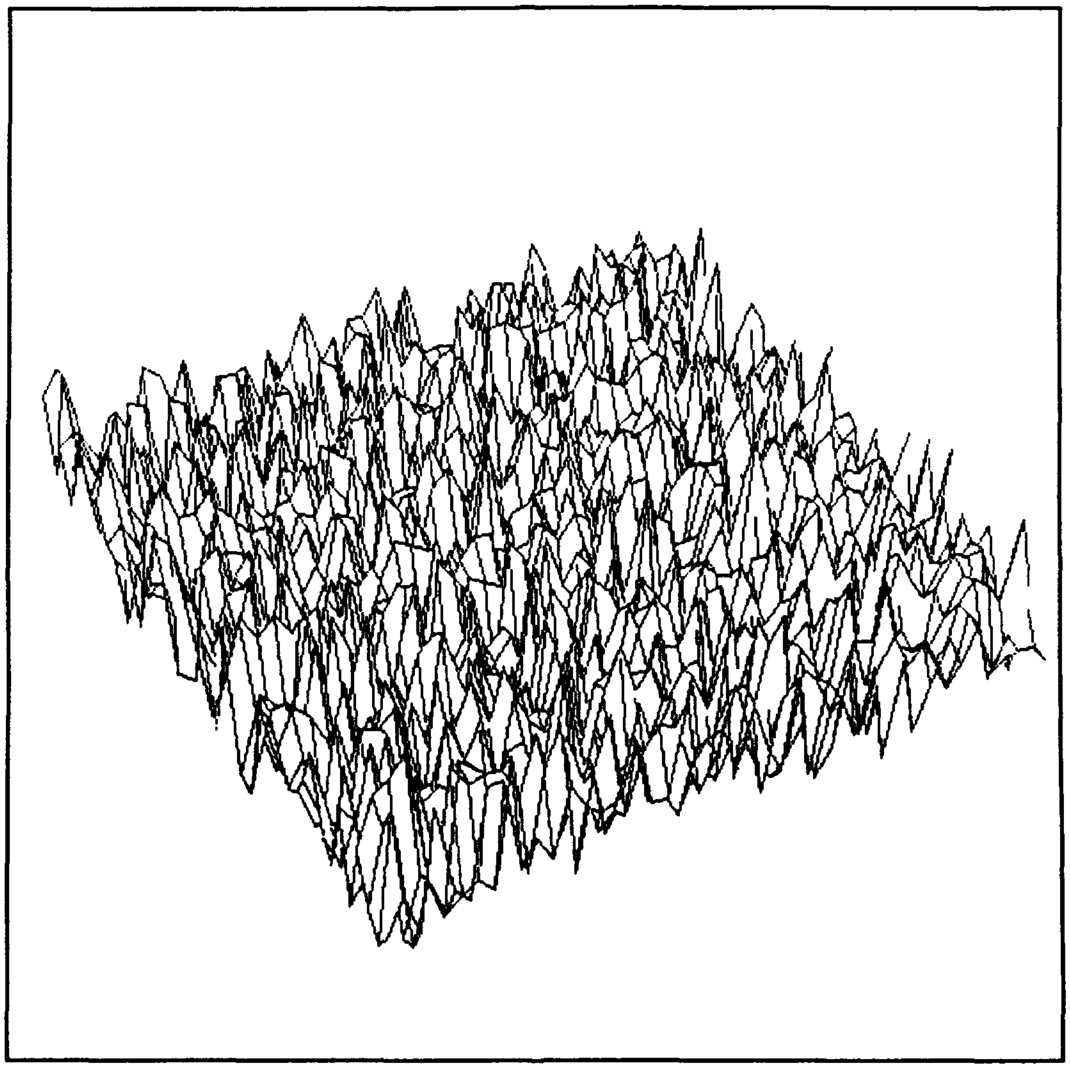

Figure 12B. Mesh Representation - Increasing

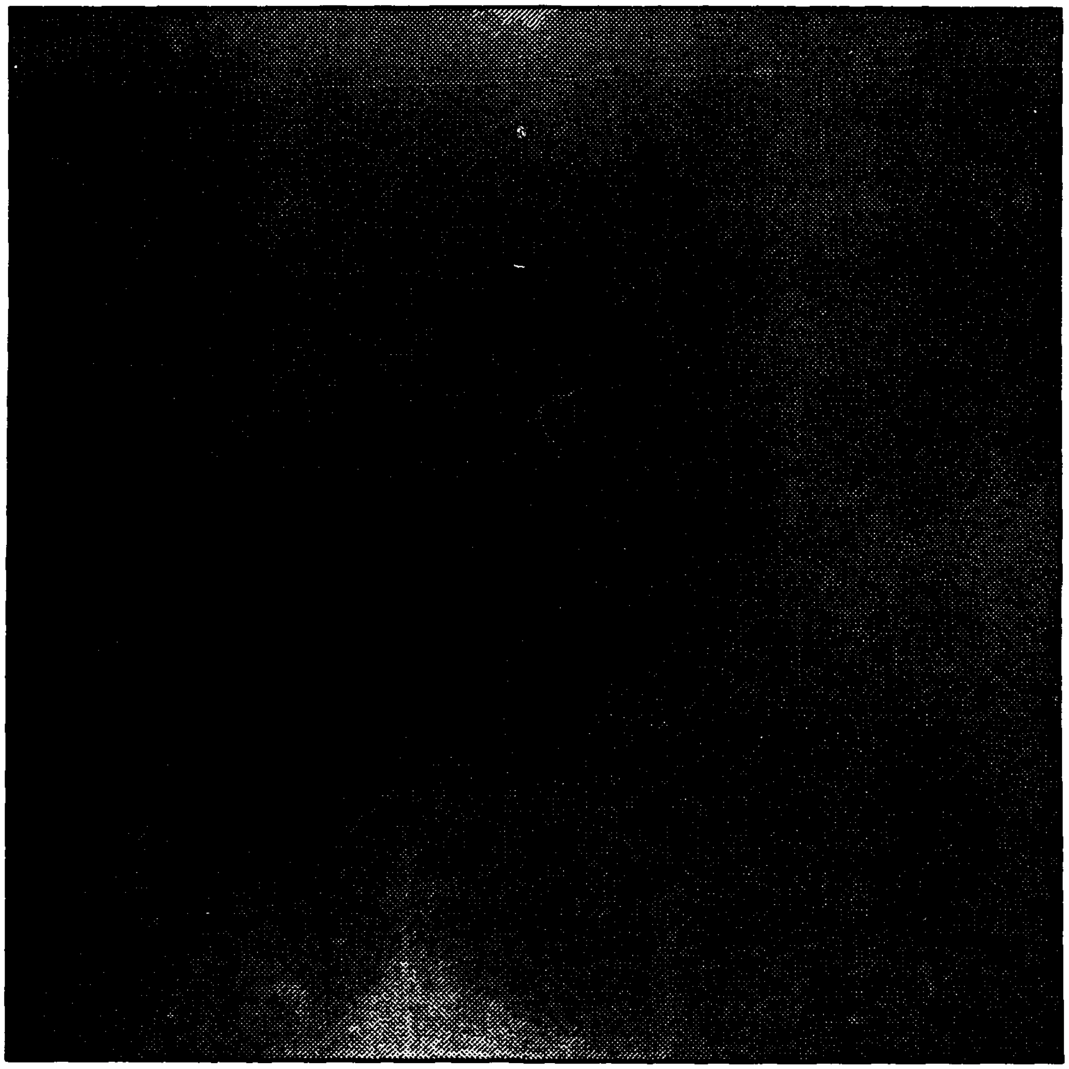

Figure 13A. Alteration - Decreasing

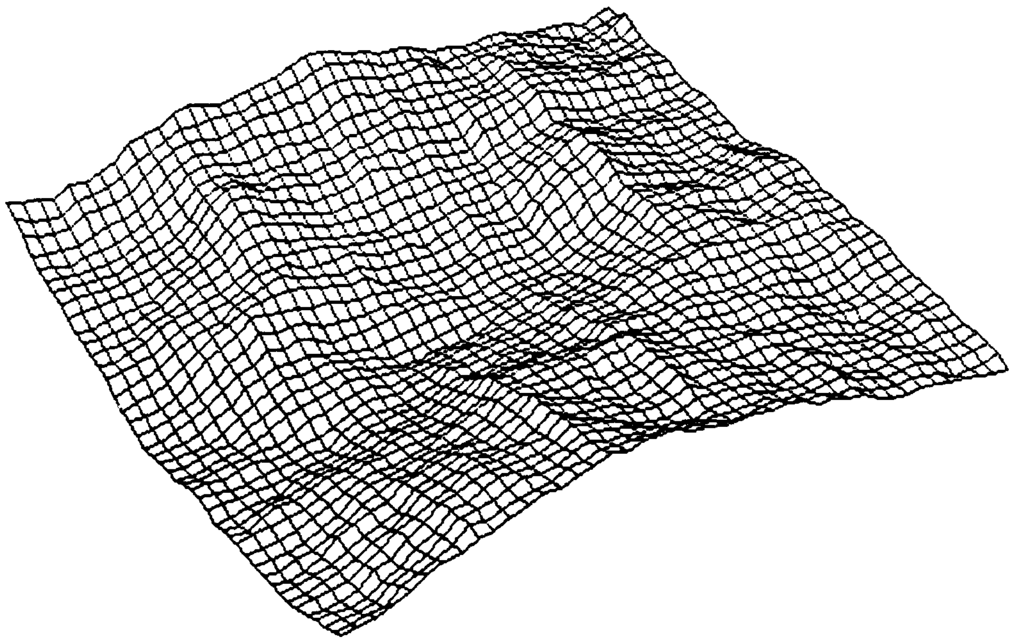

Figure 13B. Mesh Representation - Decreasing

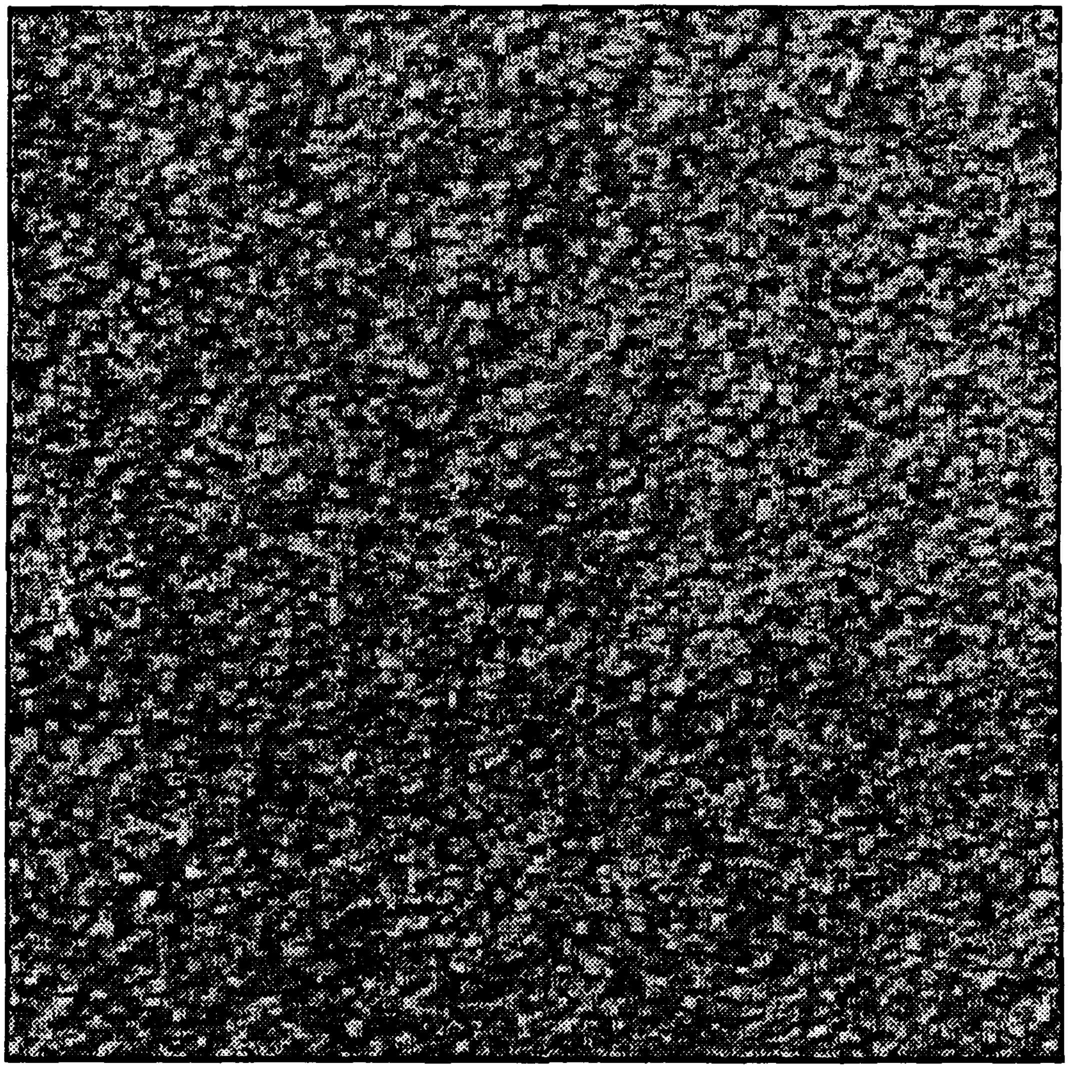

Figure 14A. Sand

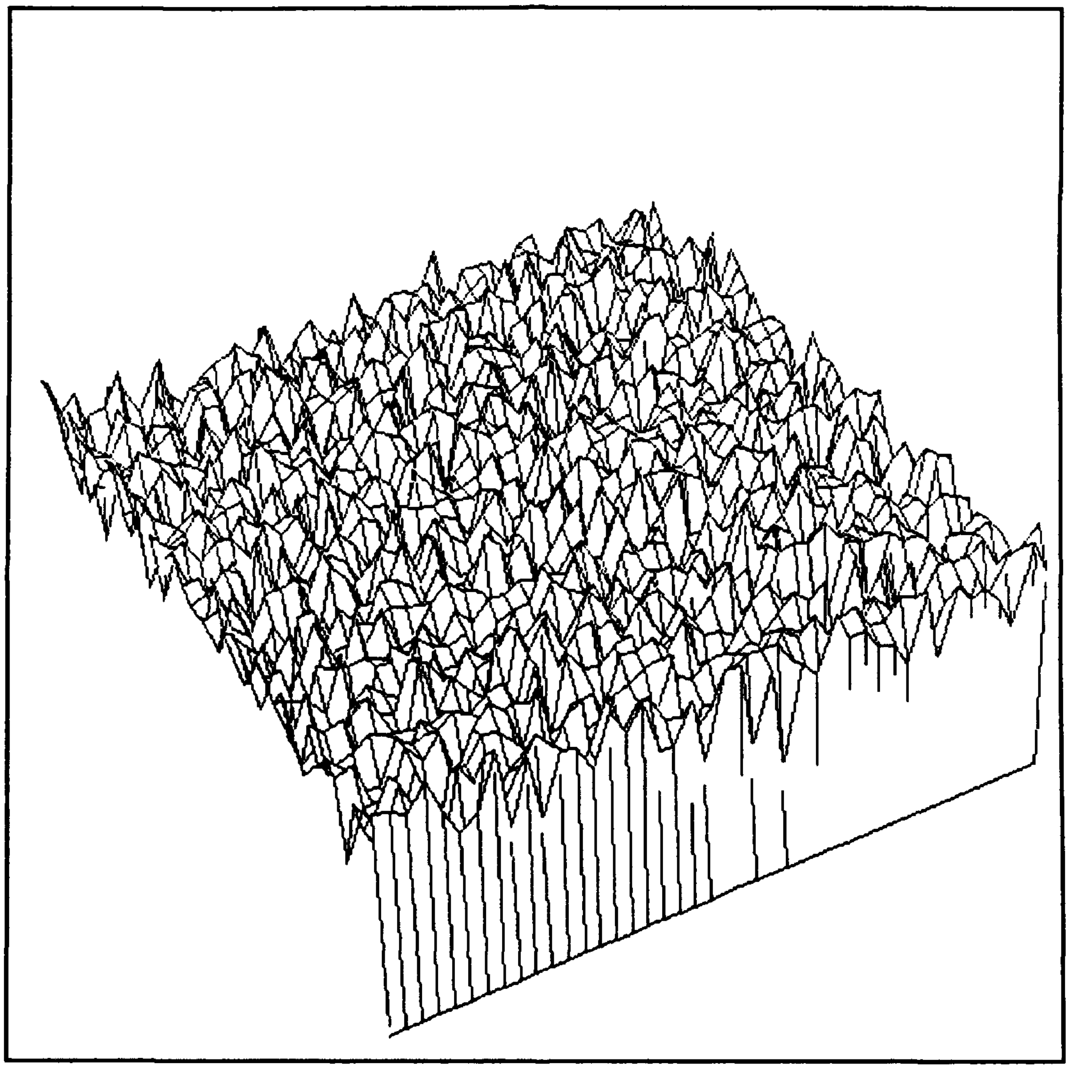

Figure 14B. Mesh Representation - Sand

Figure 15A. Rough Wall

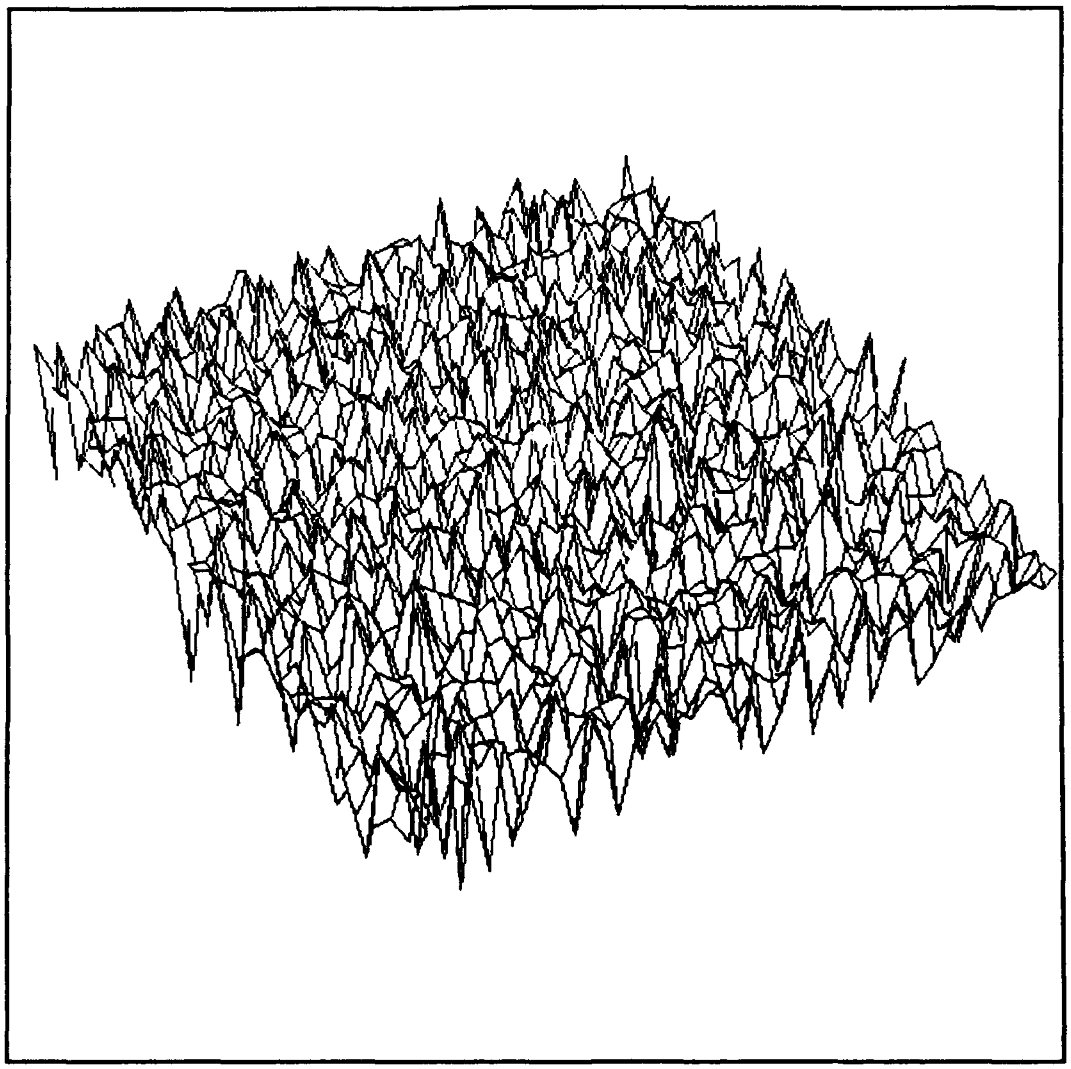

Figure 15B. Mesh Representation - Rough Wall

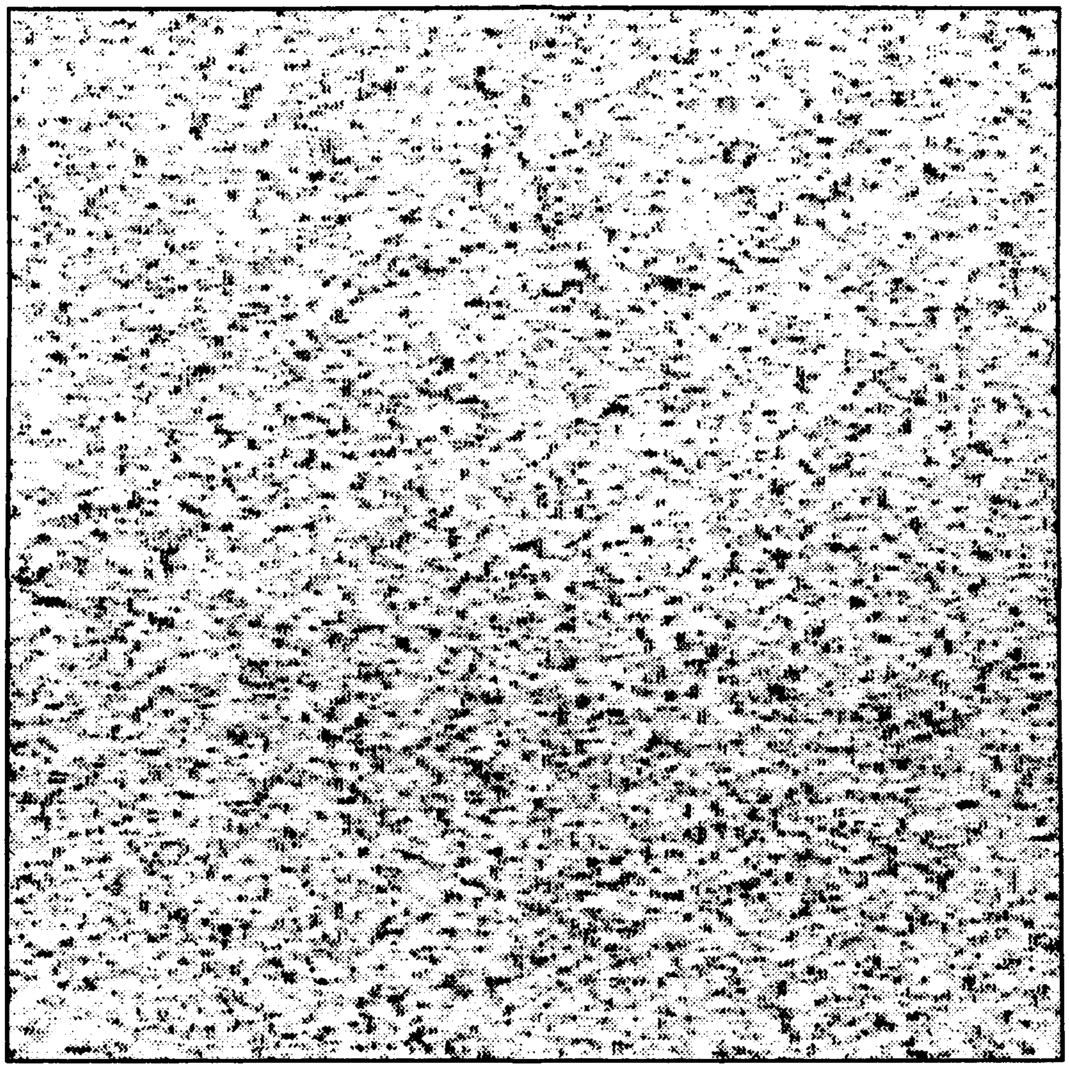

Figure 16A. Wet Sand

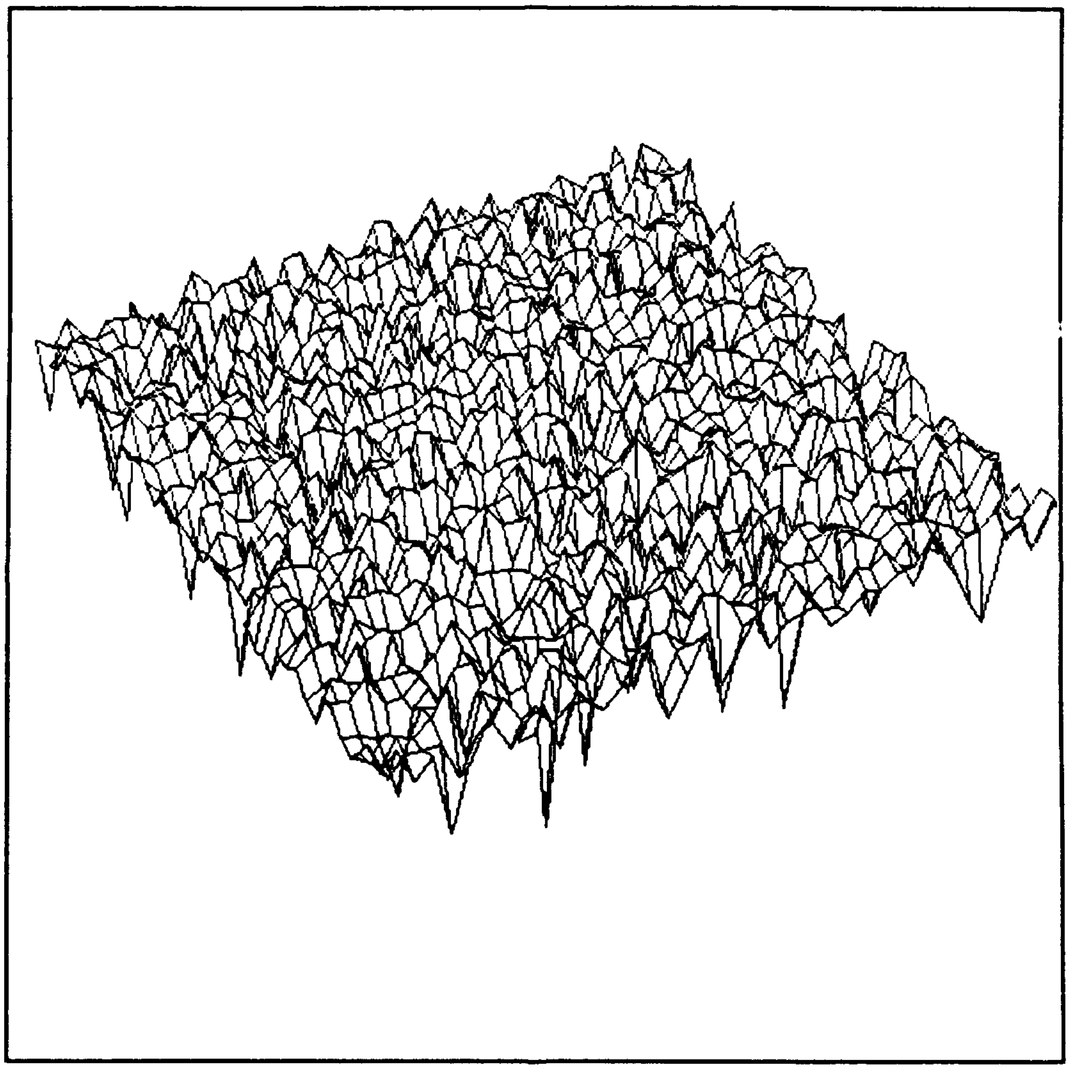

Figure 16B. Mesh Representation - Wet Sand

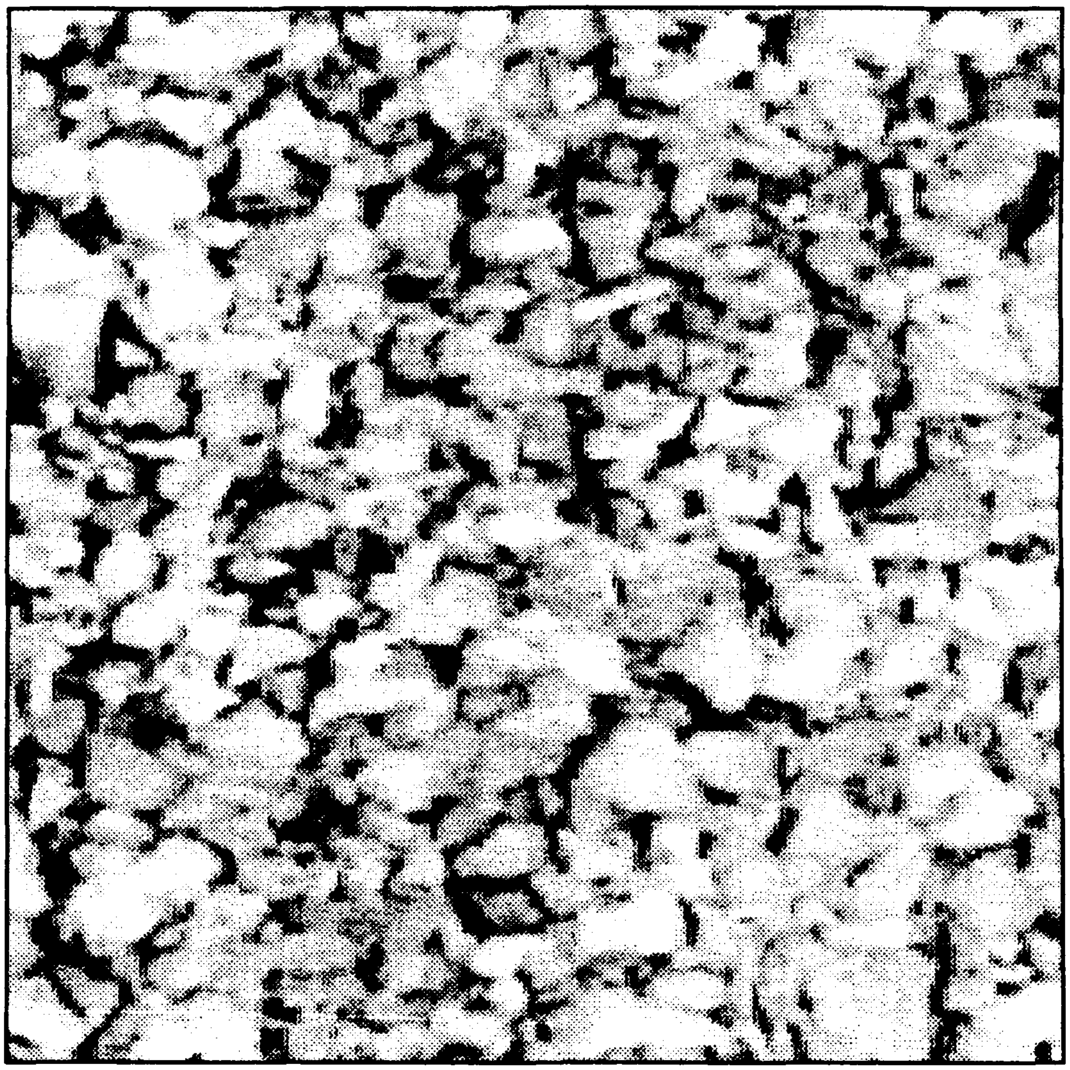

Figure 17A. Gravel

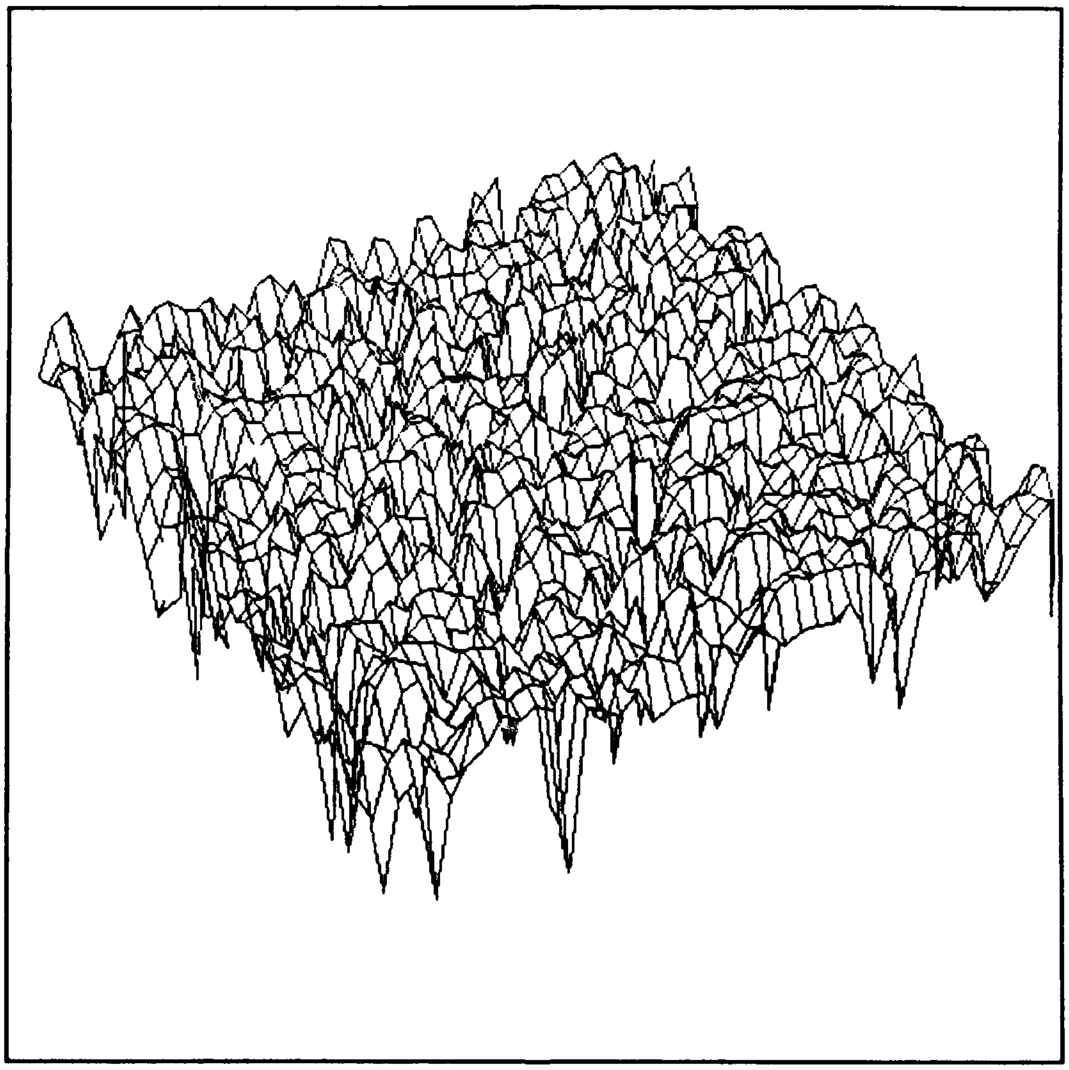

Figure 17B. Mesh Representation - Gravel

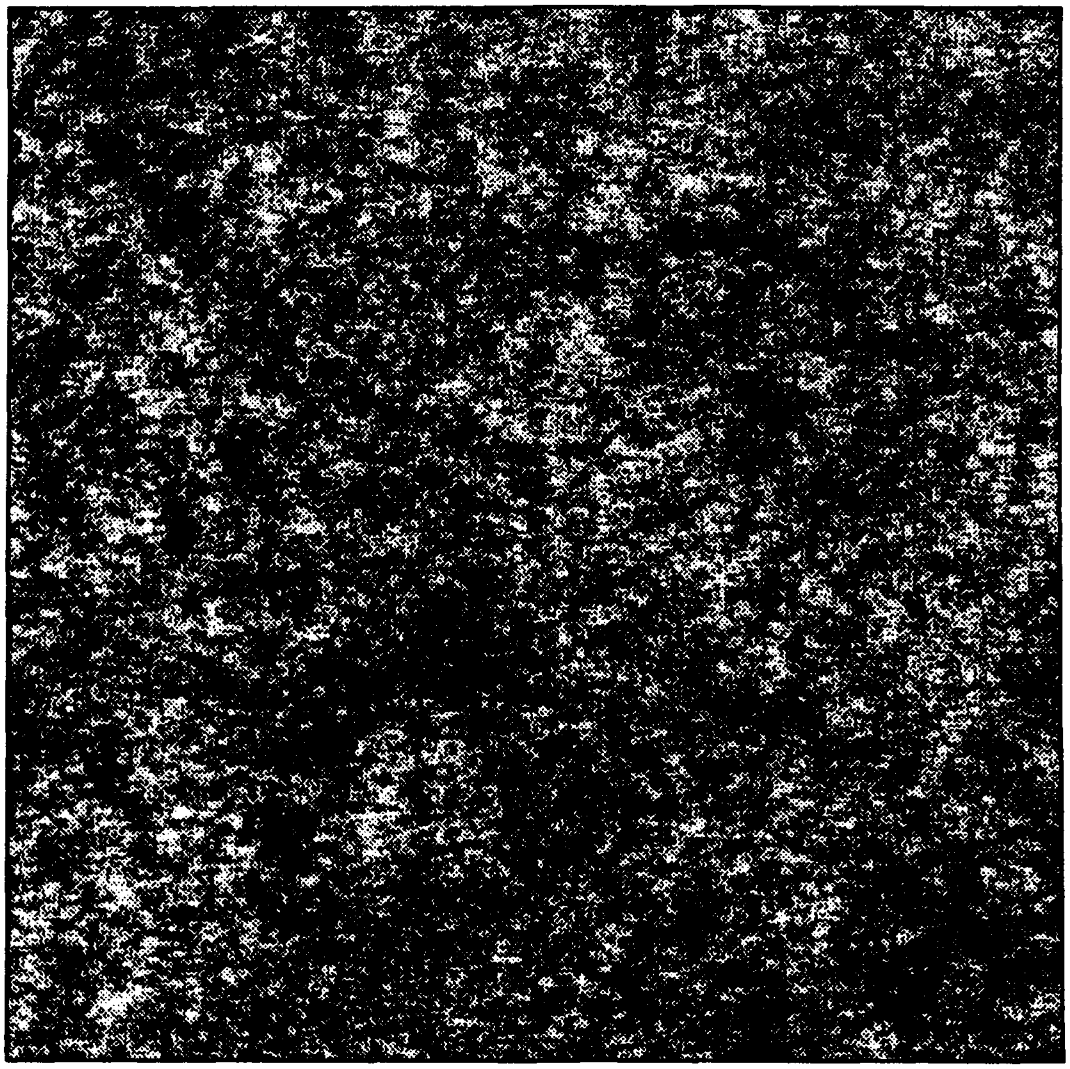

Figure 18A. Fractal Simulation of Sand

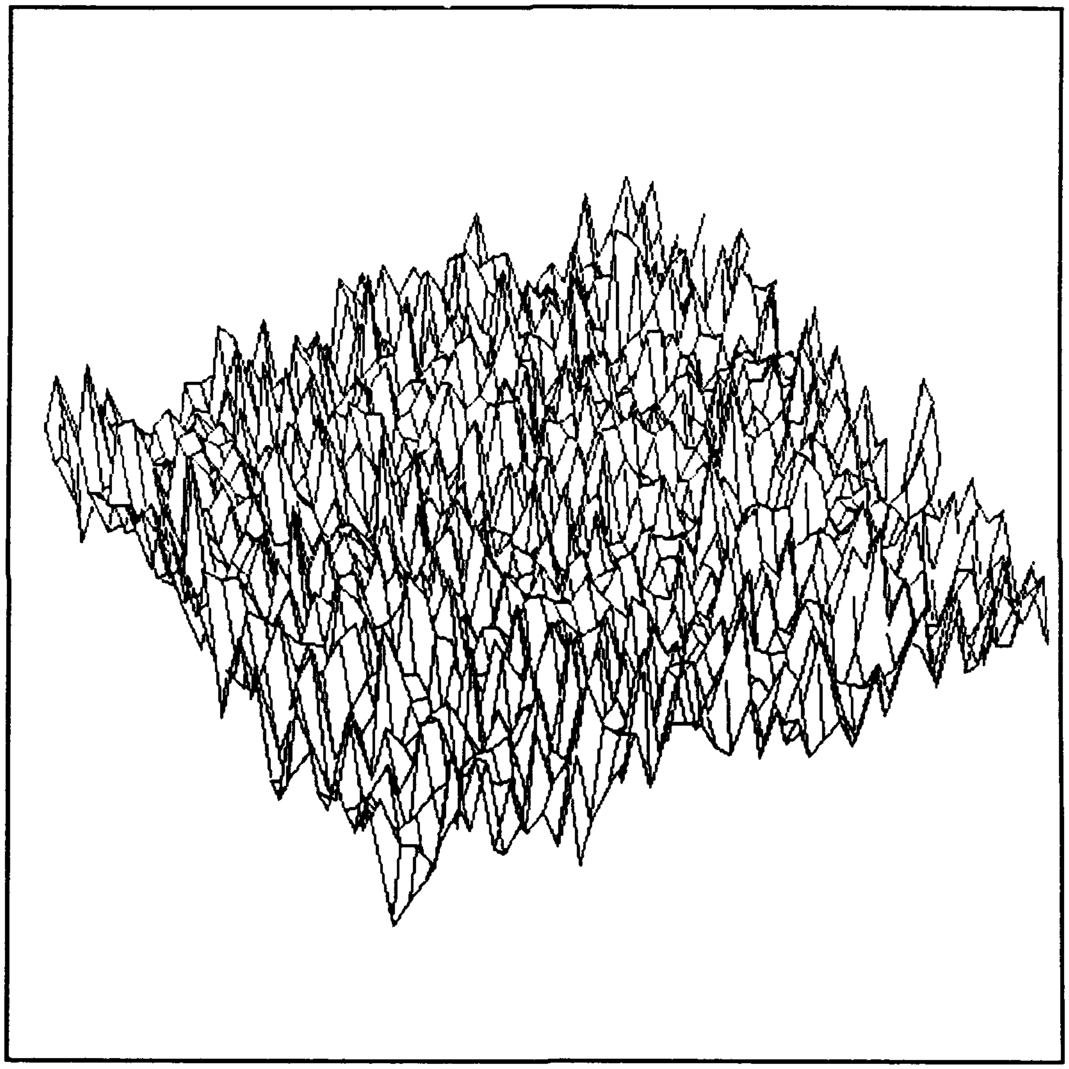

Figure 18B. Mesh Representation - Fractal Simulation of Sand

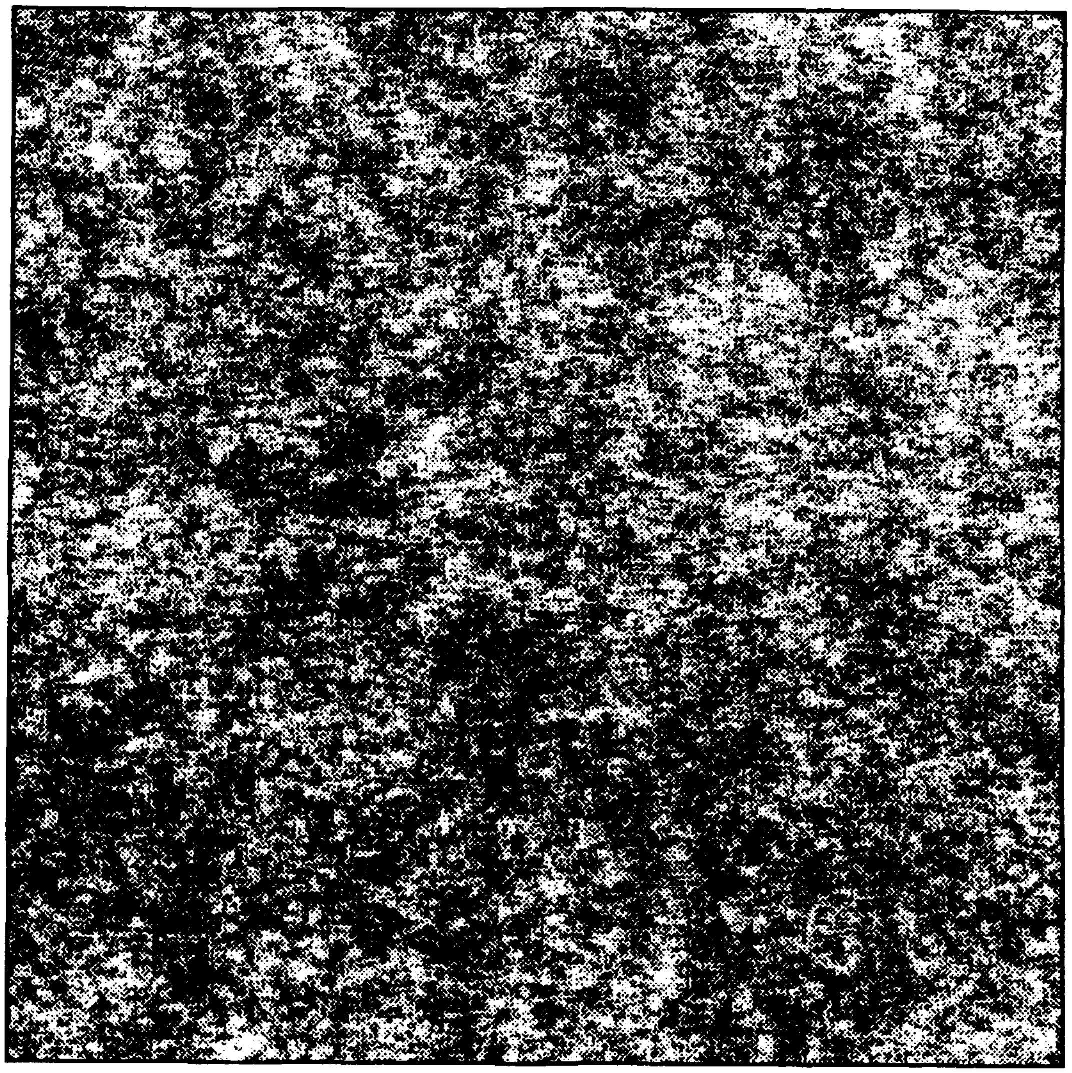

Figure 19A. Fractal Simulation of Rough Wall

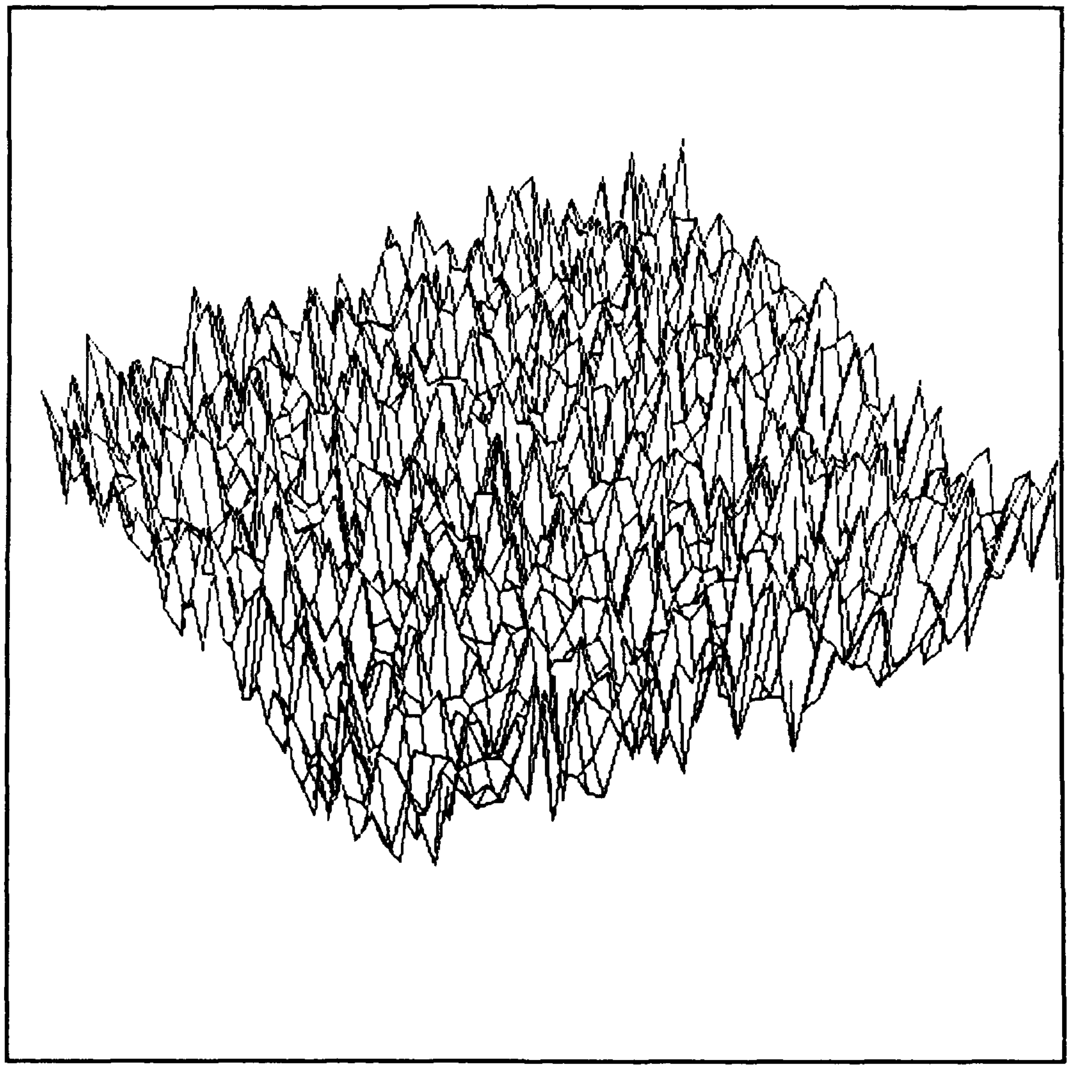

Figure 19B. Mesh Representation - Fractal Simulation of Rough Wall

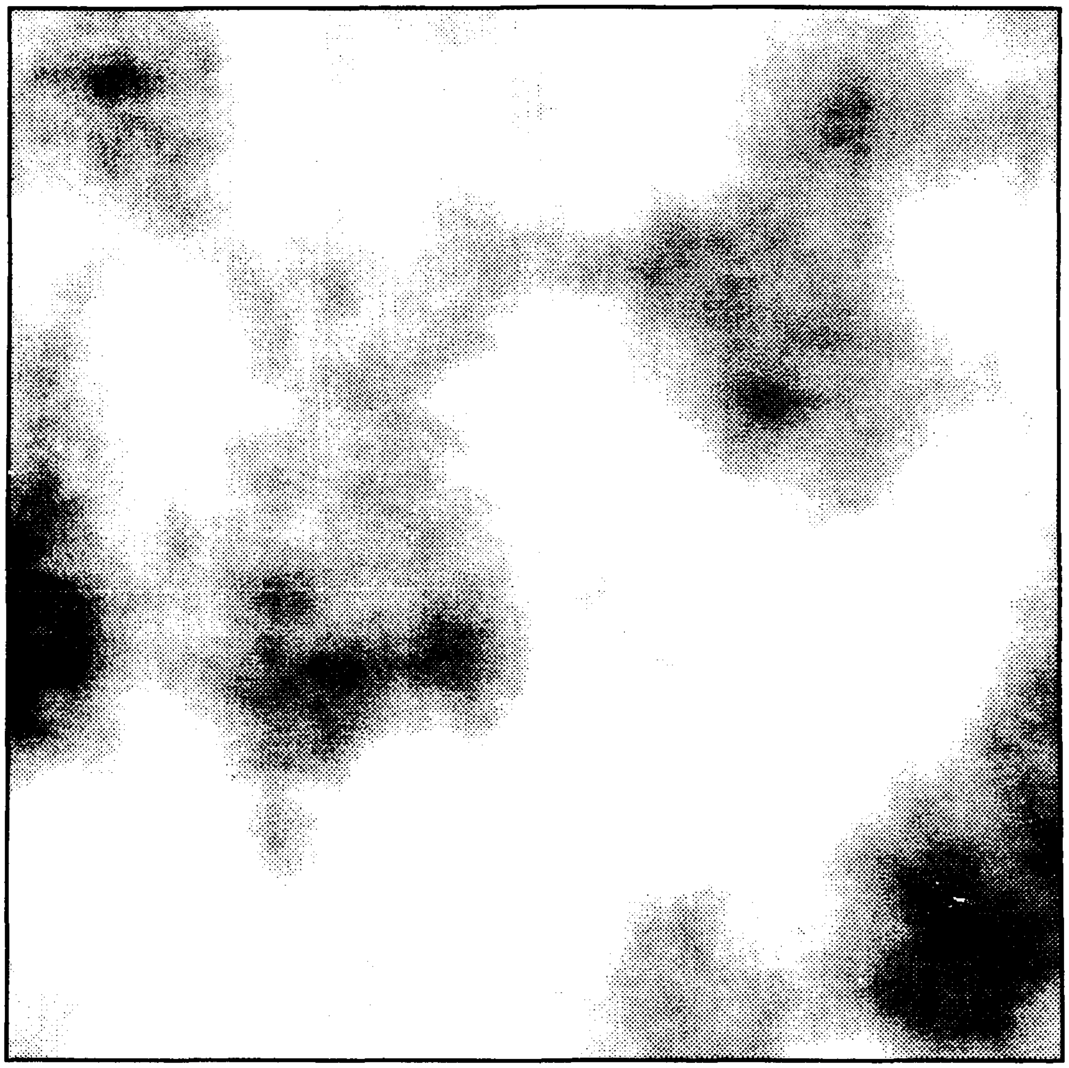

Figure 20A. Fractal Simulation of Wet Sand

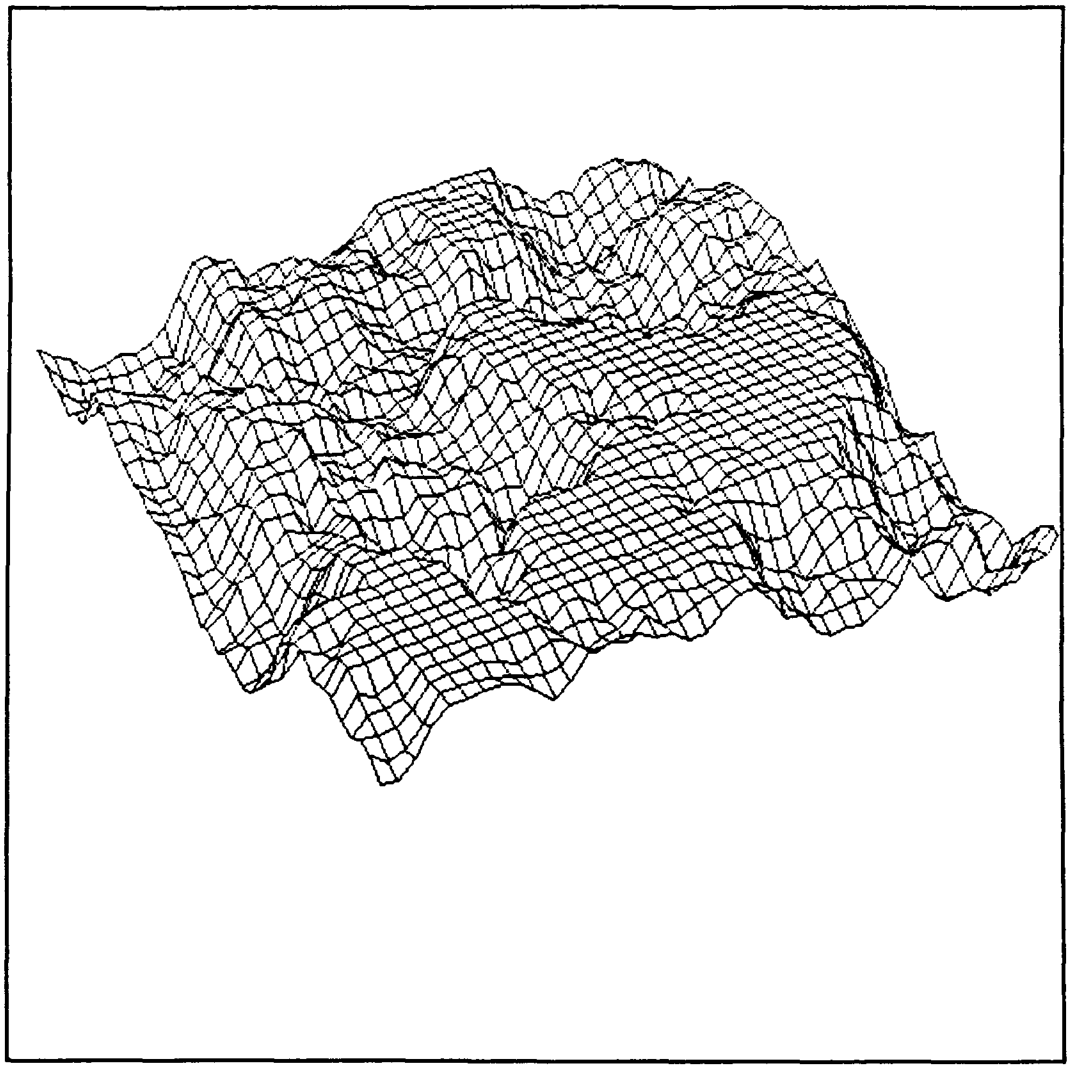

Figure 20B. Mesh Representation - Fractal Simulation of Wet Sand

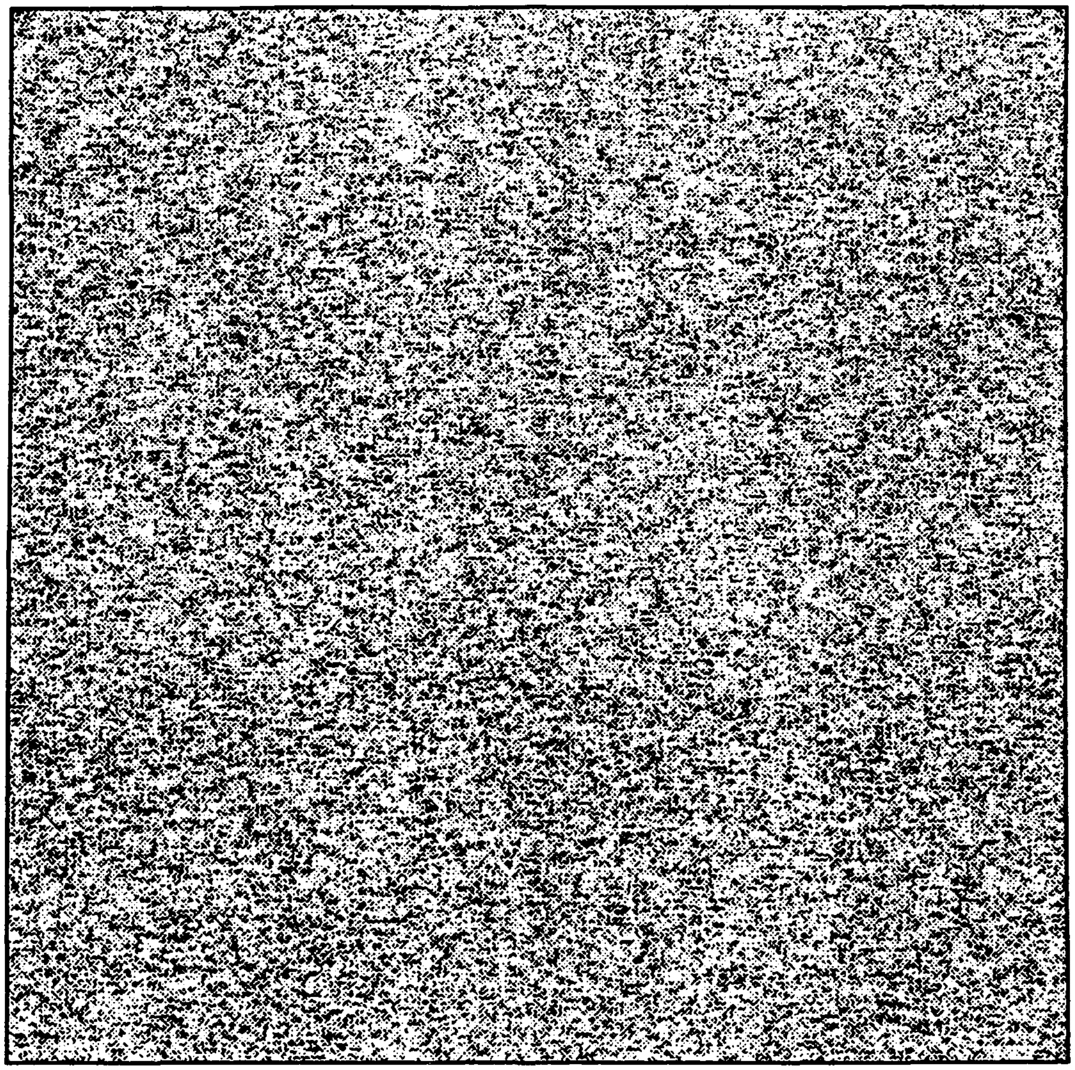

Figure 21A. Fractal Simulation of Gravel

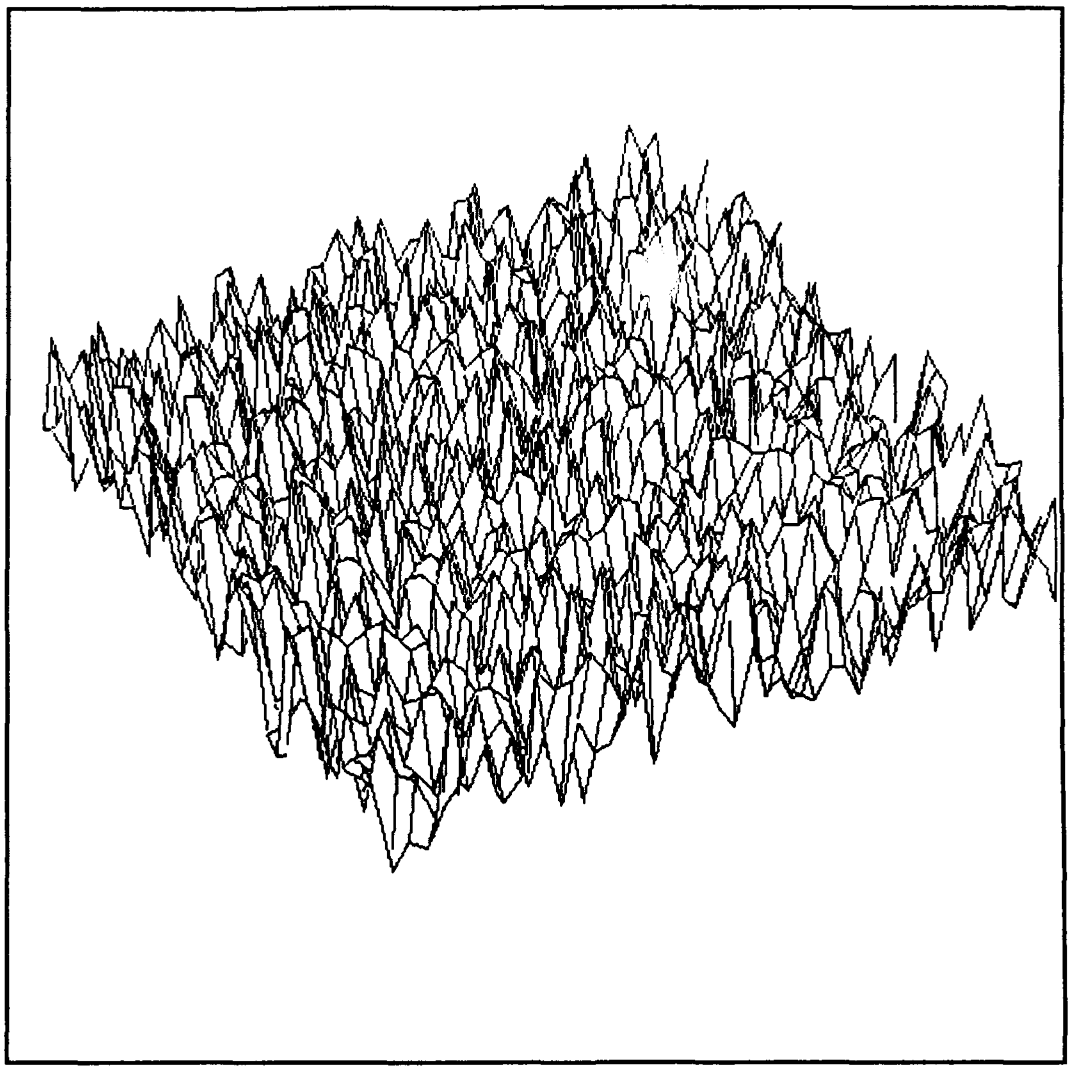

Figure 21B. Mesh Representation - Fractal Simulation of Gravel

APPENDIX B

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Module Name:    BFRACT.FOR
c       Description:    Backround simulation by fractal generation of topography
c       Authors:        C. Robertson
c       Creation Date:  5 May 89
c       Revision Date:  11 JUL 91
c
c...5....10...15...20...25...30...35...40...45...50...55...60...65...70...75...
c
        real z(1),rvar(9),tvar
        character*50 finam
        integer*4 im(513,513),ima,ar(11)
c
c       Create a fractal array to simulate the ocean bottom
c
        write(6,*) ' What value is chosen for the outer corners?'
        write(6,*) ' (1 through 255)'           ! pick initial corner values
        read(5,*) ima
        write(6,*) ' What variation values do you want?'
        write(6,*) ' Enter 9 values, singly.'
        read(5,*) rvar(1)                       ! distance of 256
        read(5,*) rvar(2)                       ! distance of 128
        read(5,*) rvar(3)                       ! distance of 64
        read(5,*) rvar(4)                       ! distance of 32
        read(5,*) rvar(5)                       ! distance of 16
        read(5,*) rvar(6)                       ! distance of 8
        read(5,*) rvar(7)                       ! distance of 4
        read(5,*) rvar(8)                       ! distance of 2
        read(5,*) rvar(9)                       ! distance of 1
        write(6,*) ' What do you want to name this file?'
        read(5,44) finam                        ! name the output file
 44     format(a50)                             ! format for file name
c
        N=9                                     ! # of steps in a 512x512 image
c
        ni=2**(N)+1                             ! defines size of fractal
c
        do 200 i=1,N
          ar(i)=2**(N-i)                        ! fill array of 2**N's
 200    continue
c
        im(1,1)=ima                             ! fill initial corner
        im(1,ni)=ima                            ! values
        im(ni,1)=ima
        im(ni,ni)=ima
c
c       iseed=0 uses system clock : iseed>0 provides a repeatable seed
        call rnset(0)                           ! system clock used as seed
c
c       LAYER nn
c
        tvar=2.                                 ! converts multiple to ±1
        do 1 nn=1,N                             ! Loops through steps in fractal
          iar=ar(nn)*2                          ! step size
          iiar=ar(nn)+1                         ! starting position
          iiiar=ni-ar(nn)                       ! ending position
          do 2 i=iiar,iiiar,iar                 ! y position
          do 2 j=1,ni,iar                       ! x position
        call rnun(1,z)                          ! call 1 random number
```

```
          im(j,i)=(im(j,i+ar(nn))+im(j,i-ar(nn)))/2.+
     &    ((z(1)-.5)*rvar(nn)*tvar)              ! calculate pixel value
 2      continue                                 ! loop
c
        do 3 i=1,ni,iar                          ! y position
        do 3 j=iiar,iiiar,iar                    ! x position
      call rnun(1,z)                             ! call 1 random number
          im(j,i)=(im(j+ar(nn),i)+im(j-ar(nn),i))/2.+
     &    ((z(1)-.5)*rvar(nn)*tvar)              ! calculate pixel value
 3      continue                                 ! loop
c
        do 4 i=iiar,iiiar,iar                    ! y position
        do 4 j=iiar,iiiar,iar                    ! x position
      call rnun(1,z)                             ! call 1 random number
          im(j,i)=(im(j+ar(nn),i)+im(j-ar(nn),i)+im(j,i+ar(nn))
     &    +im(j,i-ar(nn)))/4.+((z(1)-.5)*rvar(nn)*tvar)
 4      continue                                 ! calculate pixel value
 1    continue                                   ! loop
c
c     Transfer to array to pass back to main program
c
      do 5 i=1,ni-1                              ! y position
      do 5 j=1,ni-1                              ! x position
         if(im(j,i).gt.255) im(j,i)=255          ! clip at 255
         if(im(j,i).lt.0)   im(j,i)=0            ! clip at 0
 5    continue                                   ! loop
c
c
c     To test - write to a file for display
c
      open(unit=1,file=finam,status='new',       ! open fractal file
     & recordtype='fixed',recl=512)
c
      do 101 i=1,512                             ! pixels in row
      write(1,100)(im(j,i),j=1,512)              ! write fractal to file
 101  continue                                   ! loop
 100  format(512a1)                              ! format for fractal
c
      close(1)                                   ! close fractal file
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Name:               DIM.FOR
c       Purpose:            To determine the fractal dimension of 512 x
c                           512 images.
c       Programmer:         C. Robertson
c       Date:               13 May 91
c
c...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75
c
        real*8 sd
        real*16 msq
        integer odd(6),in(6),jn(6)
        real a(6),la(6),eps(6),d(6),leps(6)
        integer*2 image(512,512),im
        real*16 m,holder
        character*50 namin
c
        data odd/64,256,1024,4096,16384,65536/
        data in/64,32,16,8,4,2/
        data jn/64,32,16,8,4,2/
        data eps/64,32,16,8,4,2/
c
        write(6,*) ' What is the name of input file?'
        read(5,1) namin                              ! name file to be evaluated
 1      format(a50)                                  ! format of input file name
        open(unit=1,file=namin,status='old',readonly)   ! open input file
        read(1,2) image                              ! read input file
 2      format(512a1)                                ! format of input file
        m=0                                          ! clear mean
        msq=0.                                       ! clear mean sq
        do 3 i=1,512                                 ! y position
          do 4 j=1,512                               ! x position
          im=iand(image(j,i),'00ff'x)                ! clear upper byte
          holder=im+holder                           ! accumulate pixel values
          msq=float(im)*float(im)+msq                ! accumulate square of pixel
 4         continue                                  ! loop
 3      continue                                     ! loop
c
c       Find mean and standard deviation
c
        m=holder/(512.*512.)                         ! calculate mean
        sd= sqrt(msq/(512.*512.)-m*m)                ! calculate standard deviation
c
        do 100 ia=1,6
          nnn=0                                      !initialize data holder
c
          do 200 i= 1,512,in(ia)                     ! x position
            do 300 j=1,512,jn(ia)                    ! y position
              nn=0                                   !initialize data holder
              do 400 ii=0,in(ia)-1                   ! x position adjuster
                do 500 jj=0,jn(ia)-1                 ! y position adjuster
                  if((ii.eq.0).and.(jj.eq.0)) go to 500 !unique condition
                  n=abs(image(i,j)-image(i+ii,j+jj))    !find pixel difference
c                                                       ! in two positions
                  nn=nn+n                            ! accumulate data
 500            continue                             ! loop
 400          continue                               ! loop
              nnn=nnn+nn                             ! accumulate data
 300        continue                                 ! loop
 200      continue                                         ! loop
          a(ia)=float(nnn)/(float(512*512)-odd(ia))        ! find mean pixel value
 c                                                         ! difference
          if(a(ia).eq.0) then                              ! unique condition
          la(ia)=.0001                                     ! avoid value of 0
          goto 8                                           ! go to epsilon calculator
          else                                             ! other case
          endif                                            ! if completed
          la(ia)=abs(log10(a(ia)))                         ! log of pixel value
 8        leps(ia)=log10(eps(ia))                          ! log of epsilon value
          if(leps(ia).eq.0) then                           ! unique case to avoid
          d(ia)=1000000.                                   ! dividing by 0
          goto 100                                         ! this section completed
          else                                             ! other case
          endif                                            ! if completed
          d(ia)=3-(la(ia)/leps(ia))                        ! fractal dimension
 100    continue                                           ! proceed
        write(6,*) 'mean=',m,'    standard deviation =',sd
        write(6,*) '     average       epsilon        logavg         logeps
     *     dimension'
        write(6,*) a(1),eps(1),la(1),leps(1),d(1)        !print data to screen
        write(6,*) ' '
        write(6,*) a(2),eps(2),la(2),leps(2),d(2)
        write(6,*) ' '
        write(6,*) a(3),eps(3),la(3),leps(3),d(3)
        write(6,*) ' '
        write(6,*) a(4),eps(4),la(4),leps(4),d(4)
        write(6,*) ' '
        write(6,*) a(5),eps(5),la(5),leps(5),d(5)
        write(6,*) ' '
        write(6,*) a(6),eps(6),la(6),leps(6),d(6)
        stop
        end
```

REFERENCES

Canright, D., Personal communication to Robertson, C., Subject: Fractals, 1 July 1991.

Dubuc, B., and others, "Evaluating the Fractal Dimension of Surfaces," Proceedings of the Royal Society of London, A425, 1989.

Mandelbrot, B.B., Fractals - Form, Chance, and Dimension, W.H. Freeman & Company, 1977.

Mandelbrot, B.B., The Fractal Geometry of Nature, W.H.Freeman & Company, 1983.

Pietgen, H.-O., and others, The Science of Fractal Images, Springer-Verlag, 1988.

BIBLIOGRAPHY

Barnsley, M., Fractals Everywhere, Academic Press, 1988.

Canright, D., Personal communication to Robertson, C., Subject: Fractals, 1 July 1991.

Dubuc, B., and others, "Evaluating the Fractal Dimension of Surfaces," Proceedings of the Royal Society of London, A425, 1989.

Falconer, K., Fractal Geometry - Mathematical Foundations and Applications, John Wiley and Sons, 1990.

Feder, J., Fractals, Plenum Press, 1988.

Huang, J. and Turcotte, D.L., "Fractal Mapping of Digitized Images: Application to the Topography of Arizona and Comparisons With Synthetic Images," Journal of Geophysical Research, v. 94, no. B6, pp. 7491-7495, 10 June 1989.

Jullien, R. and Botet, R., Aggregation and Fractal Aggregates, World Scientific, 1987.

Kaye, B.H., A Random Walk Through Fractal Dimensions, VCH Publishers, 1989.

Keller, J.M., Crownover, R.M., and Chen, R.Y., Characteristics of Natural Sciences Related to the Fractal Dimensions, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. PAM 1-9, no. 5, pp 621-627, September 1987.

Mandelbrot, B.B., Fractals - Form, Chance, and Dimension, W.H. Freeman & Company, 1977.

Mandelbrot, B.B., The Fractal Geometry of Nature, W.H. Freeman & Company, 1983.

McGuire, M., An Eye for Fractals, Addison-Wesley Publishing Company, 1991.

Pentland, A.P., Fractal Based Description of Natural Sciences, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. PAM 1-6, no. 6, pp 661-674, November 1984.

Pietgen, H.-O., and others, The Science of Fractal Images, Springer-Verlag, 1988.

Pietgen, H.-O., and Richter, P.H., The Beauty of Fractals - Images of Complex Dynamical Systems, Springer-Verlag, 1986.

Robertson, C.J., Smith, C.M., and Tuovila, S.M., "Fractal Based Imaging," paper presented at the Southeastern Simulation Conference, Pensacola, Florida, 15 - 18 October 1989.

Robertson, C.J., Smith, C.M., and Tuovila, S.M., "Simulating Acoustic Backgrounds with Fractal Images" NCSC-TN-976-89

Robertson, C.J., Smith, C.M., and Tuovila, S.M., "Simulating Topographic Features With Fractals," presented at the IEEE SoutheastCon'90, New Orleans, Louisiana, 1 - 4 April 1990.

Stanley, H.E. and Ostrowsky, N., On Growth and Form - Fractal and Non-Fractal Patterns in Physics, Martinus Nijhoff Publishers, 1986.

Weber, A.G., "University of Southern California SIPI Report 101 - Image Data Base," USC Signal and Image Processing Institute, June 1988.

Yokoya, N. and Yamamoto, K., "Fractal-Based Analysis ana Interpolating of 3D Natural Surface Shapes and their Application to Terrain Modeling," Computer Vision, Graphics, and Image Processing, v. 46, pp 284-302, 1989.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Library, Naval Coastal Systems Center Panama City, Florida 32407-5000	1
4.	Candace Robertson Code 2230 Naval Coastal Systems Center Panama City, Florida 32407-5000	1